Domain - 1 SDLC Automation

- Invoke an AWS Lambda function in a pipeline in CodePipeline
- Jenkins

TODO

BLOGS

GOOD LINKS:

SDLC Automation

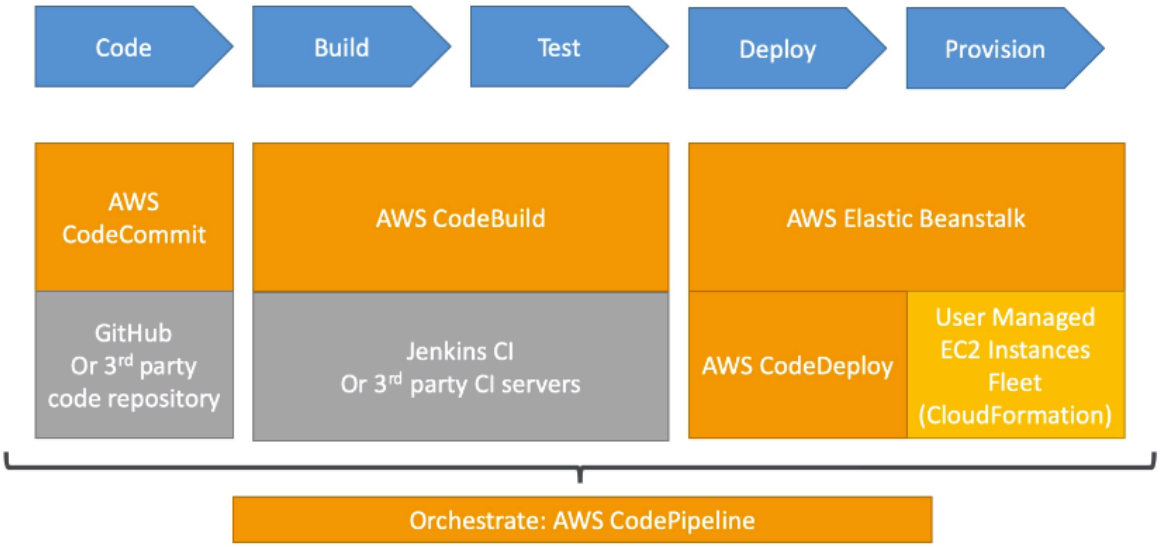
SDLC Automation

Continuous Delivery vs Continuous Deployment

- Continuous Delivery:
  - Ability to deploy often using automation
  - May involve a manual step to "approve" a deployment
  - The deployment itself is still automated and repeated!

- Continuous Deployment:
  - Full automation, every code change is deployed all the way to production
  - No manual intervention of approvals

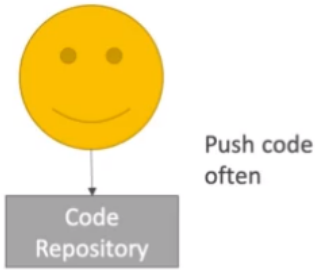
Technology Stack for CICD



CodeCommit

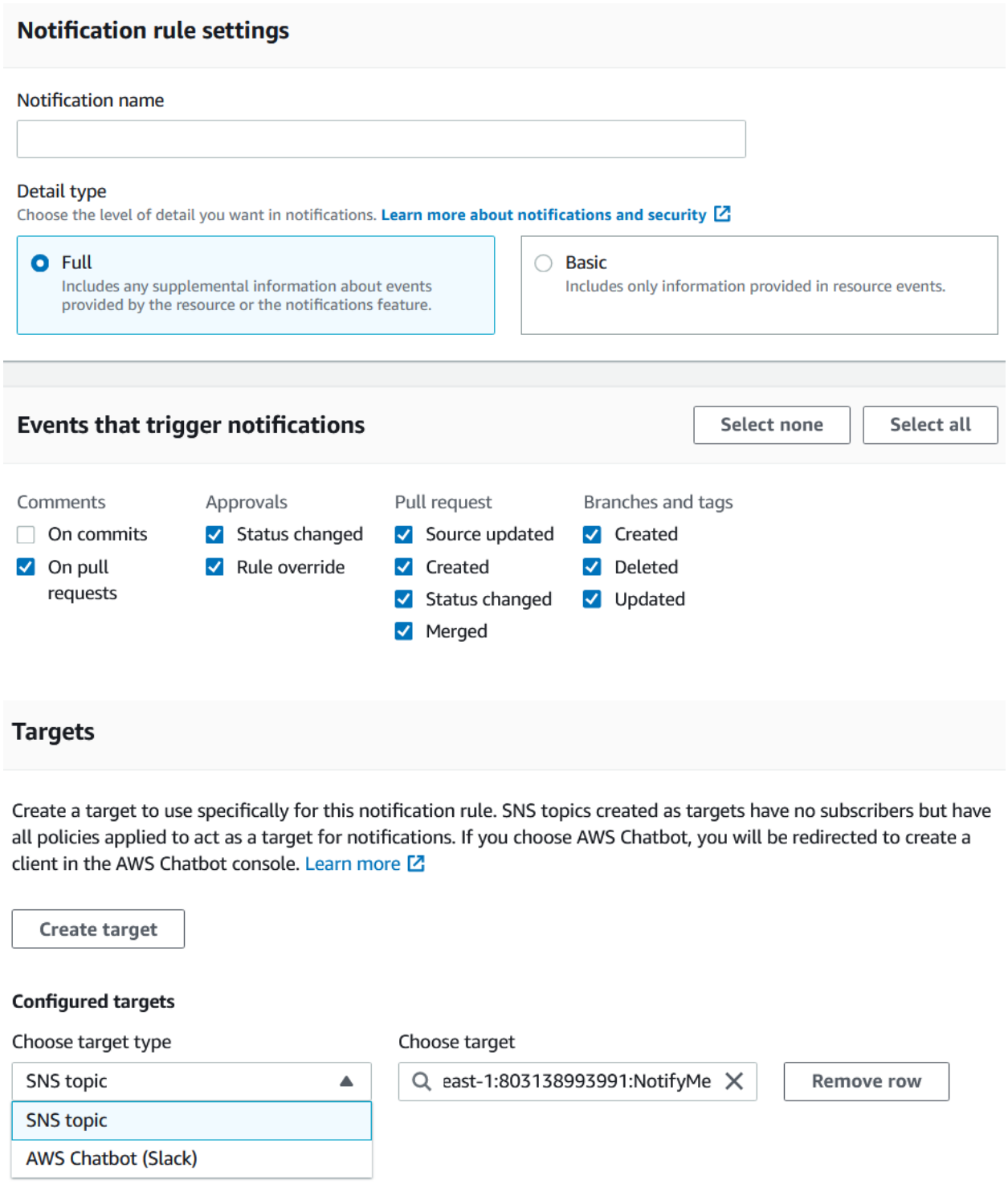
CodeCommit

- Git repositories can be expensive.
- The industry includes:
  - GitHub: free public repositories, paid private ones
  - BitBucket
  - Etc...
- And AWS CodeCommit:
  - private Git repositories
  - No size limit on repositories (scale seamlessly)
  - Fully managed, highly available
  - Code only in AWS Cloud account => increased security and compliance
  - Secure (encrypted, access control, etc…)
  - Integrated with Jenkins / CodeBuild / other CI tools



Configuring notifications for events in an AWS CodeCommit repository

Set up notification rules for a repository so that repository users receive emails about the repository event types you specify. Notifications are sent when events match the notification rule settings. You can create an Amazon SNS topic to use for notifications or use an existing one in your Amazon Web Services account.



Although you can configure a trigger to use Amazon SNS to send emails about some repository events, those events are limited to operational events, such as creating branches and pushing code to a branch. Triggers do not use CloudWatch Events rules to evaluate repository events. They are more limited in scope.

Notifications configured before November 5, 2019 had fewer event types available, and could not be configured for integration with Amazon Chime chat rooms or Slack channels. You can continue to use notifications configured before November 5, 2019, but you cannot create notifications of this type. Instead, create and use notification rules.

Share a AWS CodeCommit repository (Own Account)

When you create a repository in CodeCommit, two endpoints are generated: one for HTTPS connections and one for SSH connections. Both provide secure connections over a network. Your users can use either protocol. Both endpoints remain active regardless of which protocol you recommend to your users.

HTTPS connections require either:

- Git credentials, which IAM users can generate for themselves in IAM. Git credentials are the easiest method for users of your repository to set up and use.

- An AWS access key or role to assume, which your repository users must configure in their credential profile. You can configure **git-remote-codecommit** (recommended) or the credential helper included in the AWS CLI. These are the only methods available for root account or federated users.

SSH connections require your users to:

- Generate a public-private key pair.

- Store the public key.

- Associate the public key with their IAM user.

- Configure their known hosts file on their local computer.

- Create and maintain a config file on their local computers.

Because this is a more complex configuration process, we recommend that you choose HTTPS and Git credentials for connections to CodeCommit.

AWS provides three managed policies in IAM for CodeCommit.

AWSCodeCommitFullAccess

AWSCodeCommitPowerUser

AWSCodeCommitReadOnly

Steps:

- Create a policy (import from existing managed policies).
- Create a Group
- Attach the policy to the group
- Add users to the group
- Let the user generate the git credentials from IAM page

Configure cross-account access to an AWS CodeCommit repository

You can configure access to CodeCommit repositories for IAM users and groups in another AWS account. This is often referred to as *cross-account access*.

To configure cross-account access:

- The administrator in AccountA signs in as an IAM user with the permissions required to create and manage repositories in CodeCommit and create roles in IAM. If you are using managed policies, apply IAMFullAccess and AWSCodeCommitFullAccess to this IAM user.

  The example account ID for AccountA is 111122223333.

- The administrator in AccountB signs in as an IAM user with the permissions required to create and manage IAM users and groups, and to configure policies for users and groups. If you are using managed policies, apply IAMFullAccess to this IAM user.

  The example account ID for AccountB is 888888888888.

- The repository user in AccountB, to emulate the activities of a developer, signs in as an IAM user who is a member of the IAM group created to allow access to the CodeCommit repository in AccountA. This account must be configured with:

  - **AWS Management Console access.**

  - **An access key and secret key to use when connecting to AWS resources and the ARN of the role to assume when accessing repositories in**

AccountA.

- The **git-remote-codecommit** utility on the local computer where the repository is cloned. This utility requires Python and its installer, pip. You can download the utility from **git-remote-codecommit** on the Python Package Index website.

To allow users or groups in AccountB to access a repository in AccountA, an AccountA administrator must:

- Create a policy in AccountA that grants access to the repository.
- Create a role in AccountA that can be assumed by IAM users and groups in AccountB.
- Attach the policy to the role.

Policy

```
{
      "Version": "2012-10-17",
      "Statement": [
      {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                  "codecommit:CreateBranch",
                  "codecommit:GetTree",
                  "codecommit:GetBlob",
                  "codecommit:GetReferences",
                  "codecommit:CreateCommit",
                  "codecommit:GetPullRequestApprovalStates",
                  "codecommit:DescribeMergeConflicts",
                  "codecommit:UpdatePullRequestTitle",
                  "codecommit:BatchDescribeMergeConflicts",
                  "codecommit:GetCommentsForComparedCommit",
                  "codecommit:GetCommentReactions",
                  "codecommit:GetCommit",
                  "codecommit:GetComment",
                  "codecommit:UpdateComment",
                  "codecommit:GetCommitHistory",
                  "codecommit:GetCommitsFromMergeBase",
                  "codecommit:BatchGetCommits",
                  "codecommit:DescribePullRequestEvents",
```

```
                "codecommit:UpdatePullRequestStatus",
                "codecommit:CreatePullRequest",
                "codecommit:GetPullRequest",
                "codecommit:ListBranches",
                "codecommit:GetPullRequestOverrideState",
                "codecommit:GetRepositoryTriggers",
                "codecommit:GitPull",
                "codecommit:BatchGetRepositories",
                "codecommit:GetCommentsForPullRequest",
                "codecommit:GetObjectIdentifier",
                "codecommit:CancelUploadArchive",
                "codecommit:GetFolder",
                "codecommit:BatchGetPullRequests",
                "codecommit:GetFile",
                "codecommit:GetUploadArchiveStatus",
                "codecommit:EvaluatePullRequestApprovalRules",
                "codecommit:GetDifferences",
                "codecommit:GetRepository",
                "codecommit:GetBranch",
                "codecommit:GetMergeConflicts",
                "codecommit:GetMergeCommit",
                "codecommit:GitPush",
                "codecommit:GetMergeOptions"
            ],
            "Resource": "arn:aws:codecommit:us-east-1:803138993991:*"
      },
      {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "codecommit:GetApprovalRuleTemplate",
                "codecommit:ListRepositories"
            ],
            "Resource": "*"
      }
      ]
}
```

To create a role for repository access

- In the IAM console, choose **Roles**.

- Choose **Create role**.
- Choose **Another Amazon Web Services account**.
- In **Account ID**, enter the Amazon Web Services account ID for AccountB (for example, 888888888888). Choose **Next: Permissions**.
- In **Attach permissions policies**, select the policy you created in the previous procedure (CrossAccountAccessForMySharedDemoRepo). Choose **Next: Review**.
- In **Role name**, enter a name for the role (for example, MyCrossAccountRepositoryContributorRole). You can also enter an optional description to help others understand the purpose of the role.
- Choose **Create role**.
- Open the role you just created, and copy the role ARN (for example, arn:aws:iam::111122223333:role/MyCrossAccountRepositoryContributorRole). You need to provide this ARN to the AccountB administrator.

The simplest way to manage which IAM users in AccountB can access the AccountA repository is to create an IAM group in AccountB that has permission to assume the role in AccountA, and then add the IAM users to that group.

To create a group for cross-account repository access

Sign in to the AWS Management Console as an IAM user with the permissions required to create IAM groups and policies and manage IAM users in AccountB.

Open the IAM console at https://console.aws.amazon.com/iam/

- In the IAM console, choose **Groups**.
- Choose **Create New Group**.
- In **Group Name**, enter a name for the group (for example, DevelopersWithCrossAccountRepositoryAccess). Choose **Next Step**.
- In **Attach Policy**, choose **Next Step**. You create the cross-account policy in the next procedure. Finish creating the group.

- In the IAM console, choose **Groups**, and then choose the name of the group you just created (for example, DevelopersWithCrossAccountRepositoryAccess).

- Choose the **Permissions** tab. Expand **Inline Policies**, and then choose the link to create an inline policy. (If you are configuring a group that already has an inline policy, choose **Create Group Policy**.)

- Choose **Custom Policy**, and then choose **Select**.

- In **Policy Name**, enter a name for the policy (for example, AccessPolicyForSharedRepository).

- In **Policy Document**, paste the following policy. In Resource, replace the ARN with the ARN of the policy created by the administrator in AccountA

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource":
"arn:aws:iam::111122223333:role/MyCrossAccountRepositoryContributorRole"
  }
}
```

Choose the **Users** tab. Choose **Add Users to Group**, and then add the AccountB IAM users.

You cannot use SSH keys or Git credentials to access repositories in another Amazon Web Services account. AccountB users must configure their computers to use either git-remote-codecommit (recommended) or the credential helper to access the shared CodeCommit repository in AccountA. However, you can continue to use SSH keys or Git credentials when accessing repositories in AccountB.

Follow these steps to configure access using **git-remote-codecommit**. If you have not already installed **git-remote-codecommit**, download it from **git-remote-codecommit**

on the Python Package Index website.

Steps:

- ☐ Install the AWS CLI on the local computer.
- ☐ Install Git on the local computer

- [ ] From the terminal or command line, at the directory location where you want to clone the repository, run the **git config --local user.name** and **git config --local user.email** commands to set the user name and email for the commits you will make to the repository
- [ ] Run the **aws configure --profile** command to configure a default profile to use when connecting to resources in AccountB.

  Content to be added:

```
[default]
account = 888888888888
region = us-east-2
output = json
```

- [ ] Run the **aws configure --profile** command again to configure a named profile to use when connecting to the repository in AccountA. When prompted, provide the access key and secret key for your IAM user

```
[profile MyCrossAccountAccessProfile]
region = us-east-2
account = 111122223333
role_arn =
arn:aws:iam::111122223333:role/MyCrossAccountRepositoryContributorRole
source_profile = default
output = json
```

- [ ] To clone: git clone codecommit://MyCrossAccountAccessProfile@MySharedDemoRepo

To access the cross-account repository in the CodeCommit console

- [ ] Sign in to the AWS Management Console in AccountB (888888888888) as the IAM user who has been granted cross-account access to the repository in AccountA
- [ ] Choose your username on the navigation bar, and in the drop-down list, choose Switch Role

```
On the Switch Role page, do the following:
In Account, enter the account ID for AccountA ( for example, 111122223333).
In Role, enter the name of the role you want to assume for access to the
repository in AccountA (for example,
MyCrossAccountRepositoryContributorRole).
In Display Name, enter a friendly name for this role. This name appears in
the console when you are assuming this role. It also appears in the list of
```

```
assumed roles the next time you want to switch roles in the console.
(Optional) In Color, choose a color label for the display name.
Choose Switch Role.
```

- [ ] Open the CodeCommit console at https://console.aws.amazon.com/codesuite/codecommit/homeIf the assumed role has permission to view the names of repositories in AccountA, you see a list of repositories and an error message that informs you that you do not have permissions to view their status
- [ ] If the assumed role does not have permission to view the names of repositories in AccountA, you see an error message and a blank list with no repositories
- [ ] In **Code**, find the name of the file you added from your local computer

Triggers for an AWS CodeCommit

You can create up to 10 triggers for each CodeCommit repository.

Scenarios like notifying an external build system require writing a Lambda function to interact with other applications. The email scenario simply requires creating an Amazon SNS topic.

You can integrate Amazon SNS topics and Lambda functions with triggers in CodeCommit, but you must first create and then configure resources with a policy that grants CodeCommit the permissions to interact with those resources. You must create the resource in the same AWS Region as the CodeCommit repository.

For Amazon SNS topics, you do not need to configure additional IAM policies or permissions if the Amazon SNS topic is created using the same account as the CodeCommit repository. You can create the CodeCommit trigger as soon as you have created and subscribed to the Amazon SNS topic.

If you want to configure your trigger to use an Amazon SNS topic in another AWS account, you must first configure that topic with a policy that allows CodeCommit to publish to that topic

You can configure Lambda functions by creating the trigger in the Lambda console as part of the function. This is the simplest method, because triggers created in the Lambda console automatically include the permissions required for CodeCommit to invoke the Lambda function. If you create the trigger in CodeCommit, you must include a policy to allow CodeCommit to invoke the function.

Create a trigger to an Amazon SNS topic for a CodeCommit repository

- [ ] If you choose **All repository events**, you cannot choose any other events. To choose a subset of events, remove **All repository events**, and then choose one or more events from the list. For example, if you want the trigger to run only when a user creates a branch or tag in the CodeCommit repository, remove **All repository events**, and then choose **Create branch or tag**.
- [ ] If you want the trigger to apply to all branches of the repository, in **Branches**, leave the selection blank, as this default option applies the trigger to all branches automatically. If you want this trigger to apply to specific branches only, choose up to 10 branch names from the list of repository branches.
- [ ] Amazon SNS FIFO (first in, first out) topics are not supported for CodeCommit triggers. You must choose an Amazon SNS topic that has its type set to Standard. If you want to use an Amazon SNS FIFO topic, you must configure an Amazon Eventbridge rule for CodeCommit events that has the SNS FIFO topic configured as its target.

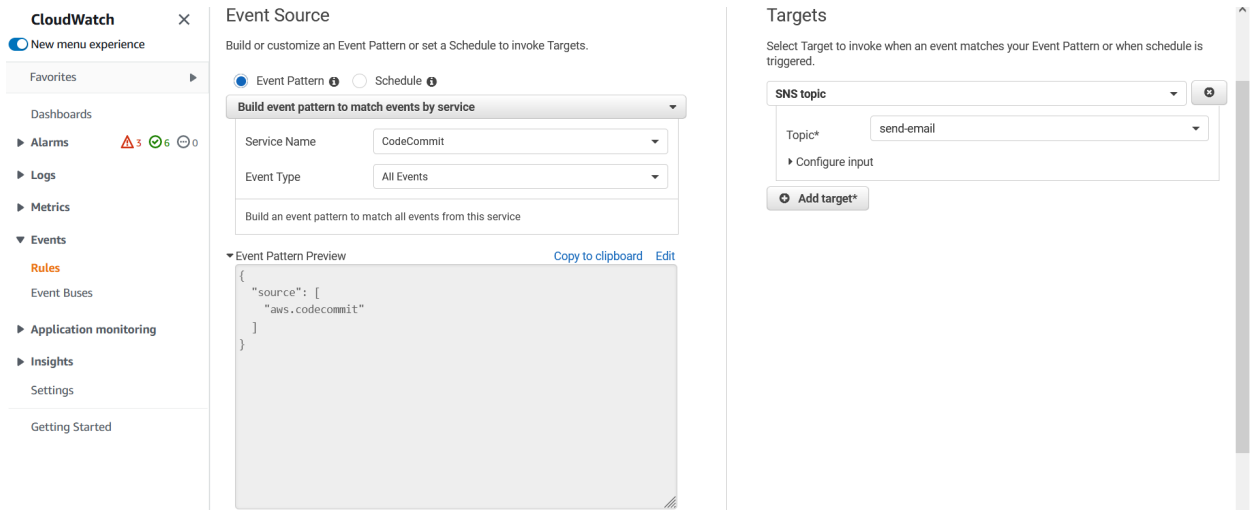
Create an AWS CodeCommit trigger for an AWS Lambda function

- [ ] Resource Policy for Lambda Trigger

```
{
  "Version": "2012-10-17",
  "Id": "default",
  "Statement": [
      {
      "Sid": "lambda-e2612f89-50bc-409f-92cd-a26ae6806075",
      "Effect": "Allow",
      "Principal": {
      "Service": "codecommit.amazonaws.com"
      },
      "Action": "lambda:InvokeFunction",
      "Resource":
"arn:aws:lambda:us-east-1:803138993991:function:EventCodeCommit-PushOrderSe
rvice",
      "Condition": {
      "ArnLike": {
            "AWS:SourceArn":
"arn:aws:codecommit:us-east-1:803138993991:OrderService"
      }
      }
      }
  ]
}
```

- [ ] The easiest way to create a trigger that invokes a Lambda function is to create that trigger in the Lambda console. This built-in integration ensures that CodeCommit has the permissions required to run the function.

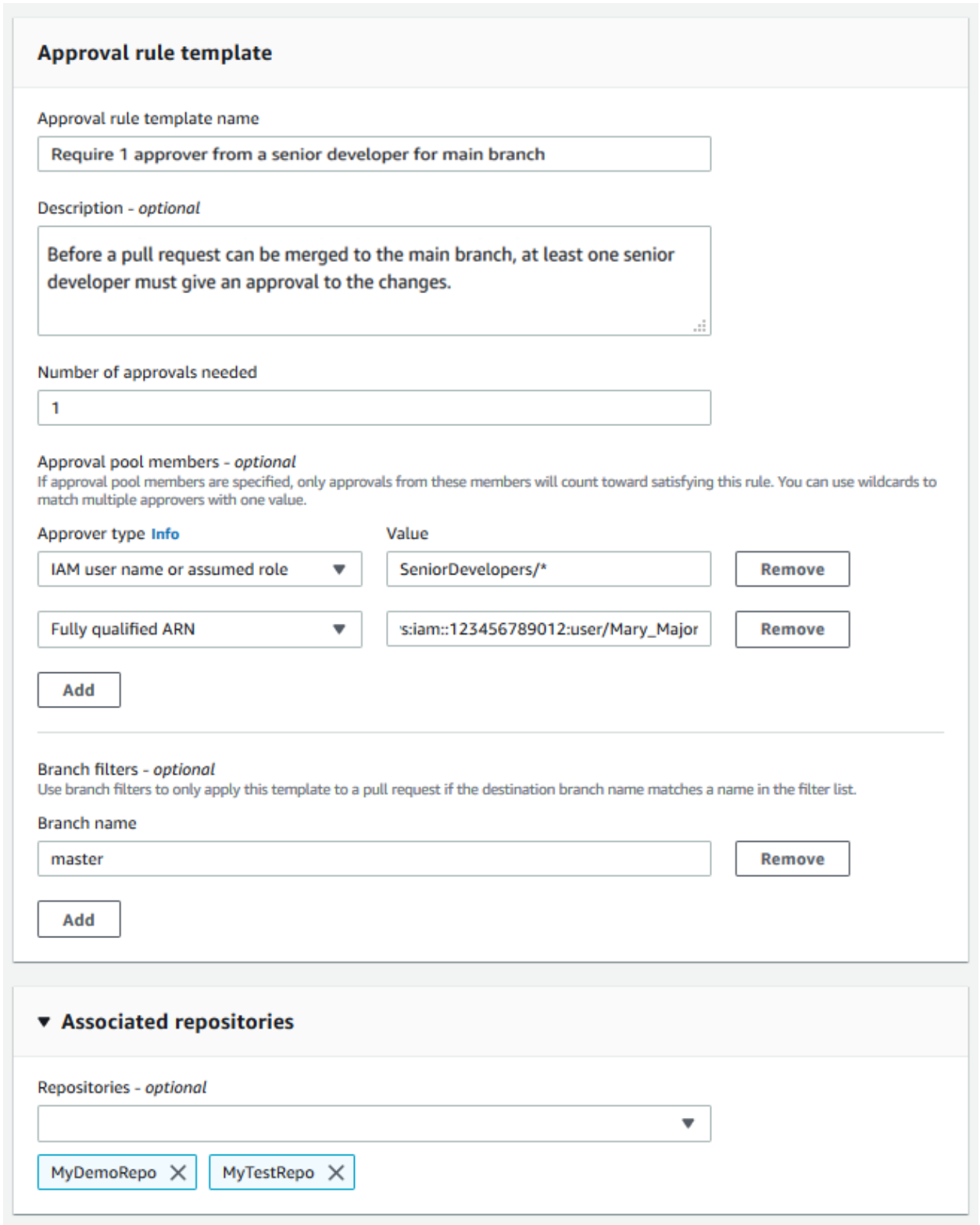
Cloudwatch Event to Trigger CodeCommit Notification



Approval Rule Templates

You can create approval rules for pull requests. To automatically apply approval rules to some or all of the pull requests created in repositories, use approval rule templates. Approval rule templates help you customize your development workflows across repositories so that different branches have appropriate levels of approvals and control. You can define different rules for production and development branches. Those rules are applied every time a pull request that matches the rule conditions is created.

You can associate an approval rule template with one or more repositories in the AWS Region where they are created. When a template is associated with a repository, it automatically creates approval rules for pull requests in that repository as part of creating the pull request. Just like a single approval rule, an approval rule template defines an approval rule structure, including the number of required approvals and an optional pool of users from which approvals must come. Unlike an approval rule, you can also define destination references (the branch or branches), also known as *branch filters*. If you define destination references, then only pull requests whose destination branch names match the specified branch names (destination references) in the template have rules created for them



IAM user name or assumed role: This option prepopulates the Amazon Web Services account ID for the account you used to sign in, and only requires a name. It can be used for both IAM users and federated access users whose name matches the provided name.

Fully qualified ARN: This option allows you to specify the fully qualified Amazon Resource Name (ARN) of the IAM user or role. This option also supports assumed roles used by other AWS services, such as AWS Lambda and AWS CodeBuild.

Both approver types allow you to use wildcards (*) in their values. For example, if you choose the **IAM user name or assumed role** option, and you specify CodeCommitReview/*, all users who assume the role of CodeCommitReview are counted in the approval pool. Their individual role session names count toward the required number of approvers.

Branch filters, enter destination branch names to use to filter the creation of approval rules You can use wildcards (*) in branch names to apply approval rules to all branch names that match the wildcard cases. However, you cannot use a wildcard at the beginning of a branch name. You can specify up to 100 branch names

Approval rule templates are created in a specific AWS Region, but they do not affect any repositores in that AWS Region until they are associated. To apply a template to one or more repositories, you must associate the template with the repository or repositories. You can apply a single template to multiple repositories in an AWS Region.

Security in AWS CodeCommit

- [ ] AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure.
- [ ] For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM). That way each user is given only the permissions necessary to fulfill their job duties
- [ ] Use multi-factor authentication (MFA) with each account.
- [ ] Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- [ ] Set up API and user activity logging with AWS CloudTrail.
- [ ] Use AWS encryption solutions, along with all default security controls within AWS services.
- [ ] Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.

- [ ] If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint
- [ ] Data in CodeCommit repositories is encrypted in transit and at rest
- [ ] The first time you create a CodeCommit repository in a new AWS Region in your Amazon Web Services account, CodeCommit creates an AWS owned key (the aws/codecommit key) in that same AWS Region in AWS Key Management Service (AWS KMS). This key is used only by CodeCommit (the aws/codecommit key). It is stored in your Amazon Web Services account. CodeCommit uses this AWS owned key to encrypt and decrypt the data in this and all other CodeCommit repositories within that region in your Amazon Web Services account.

  CodeCommit performs the following AWS KMS actions against the default aws/codecommit key. An IAM user does not need explicit permissions for these actions, but the user must not have any attached policies that deny these actions for the aws/codecommit key. Specifically, when you create your first repository, your IAM user must not have any of the following permissions set to deny:

  - "kms:Encrypt"
  - "kms:Decrypt"
  - "kms:ReEncrypt"
  - "kms:GenerateDataKey"
  - "kms:GenerateDataKeyWithoutPlaintext"
  - "kms:DescribeKey"

- [ ] CodeCommit uses two different approaches for encrypting data. Individual Git objects under 6 MB are encrypted using AES-GCM-256, which provides data integrity validation. Objects between 6 MB and the maximum 2 GB for a single blob are encrypted using AES-CBC-256. CodeCommit always validates the encryption context.
- [ ] Each service integrated with AWS KMS specifies an encryption context for both the encryption and decryption operations. The encryption context is additional authenticated information AWS KMS uses to check for data integrity. When specified for the encryption operation, it must also be specified in the decryption operation. Otherwise, decryption fails. CodeCommit uses the CodeCommit repository ID for the encryption context. You can use the **get-repository** command or the CodeCommit console to find the repository ID
- [ ] You can give users access to your AWS CodeCommit repositories without configuring IAM users for them or using an access key and secret key. Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user

directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider.

You can configure access for users who want or must authenticate through methods such as:

- Security Assertion Markup Language (SAML)
- Multi-factor authentication (MFA)
- Federation
- Login with Amazon
- Amazon Cognito
- Facebook
- Google
- OpenID Connect (OIDC)-compatible identity provider

IAM

Policy to Prevent Code Push to master branch:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "codecommit:GitPush",
                "codecommit:DeleteBranch",
                "codecommit:PutFile",
                "codecommit:MergePullRequestByFastForward"
            ],
            "Resource": "arn:aws:codecommit:us-east-2:80398EXAMPLE:MyDemoRepo",
            "Condition": {
                "StringEqualsIfExists": {
                    "codecommit:References": [
                        "refs/heads/master"
```

```
                    ]
                },
                "Null": {
                    "codecommit:References": false
                }
            }
        }
    ]
}
```

Todo

CodeBuild

CodeBuild Overview

- Fully managed build service
- Alternative to other build tools such as Jenkins
- Continuous scaling (no servers to manage or provision – no build queue)
- Pay for usage: the time it takes to complete the builds
- Leverages Docker under the hood for reproducible builds
- Possibility to extend capabilities leveraging our own base Docker images
- Secure: Integration with KMS for encryption of build artifacts, IAM for build permissions, and VPC for network security, CloudTrail for API calls logging

- Source Code from GitHub / CodeCommit / CodePipeline / S3…
- Build instructions can be defined in code (buildspec.yml file)
- Output logs to Amazon S3 & AWS CloudWatch Logs
- Metrics to monitor CodeBuild statistics
- Use CloudWatch Events to detect failed builds and trigger notifications
- Use CloudWatch Alarms to notify if you need "thresholds" for failures
- CloudWatch Events / AWS Lambda as a Glue
- SNS notifications

Video Notes

SR	Note
1	CodeBuild Timeout period 15 min to 8 hours
2	Examples:- https://docs.aws.amazon.com/codebuild/latest/userguide/use-case-based-samples.html
	CodeBuild Default Environment Variables CodeBuild Custom Environment Variables a. Provide directly from AWS console b. Mention in buildspec.yaml Env variables types: a. Plain text Parameters b. SSM Parameters

	Artifacts:
	
	Buildspec entry:
	`artifacts:`
	`  files:`
	`  - **/* ##→ All Files`
	
	• `Add IAM Policy is CodeBuild Service Role to upload artifacts to s3.`

	CloudWatch Event On :-

	a. CodeBuild to send notifications etc b. CodeCommit to trigger CodeBuild.
	https://aws.amazon.com/blogs/devops/validating-aws-codecommit-pull-requests-with-aws-codebuild-and-aws-lambda/

Sample buildspec.yaml:

```
version: 0.2

run-as: Linux-user-name

env:
  shell: shell-tag
  variables:
      key: "value"
      key: "value"
  parameter-store:
      key: "value"
      key: "value"
  exported-variables:
      - variable
      - variable
  secrets-manager:
      key: secret-id:json-key:version-stage:version-id
  git-credential-helper: no | yes

proxy:
  upload-artifacts: no | yes
  logs: no | yes

batch:
  fast-fail: false | true
  # build-list:
  # build-matrix:
  # build-graph:

phases:
```

```
  install:
      run-as: Linux-user-name
      on-failure: ABORT | CONTINUE
      runtime-versions:
      runtime: version
      runtime: version
      commands:
      - command
      - command
      finally:
      - command
      - command
  pre_build:
      run-as: Linux-user-name
      on-failure: ABORT | CONTINUE
      commands:
      - command
      - command
      finally:
      - command
      - command
  build:
      run-as: Linux-user-name
      on-failure: ABORT | CONTINUE
      commands:
      - command
      - command
      finally:
      - command
      - command
  post_build:
      run-as: Linux-user-name
      on-failure: ABORT | CONTINUE
      commands:
      - command
      - command
      finally:
      - command
      - command
reports:
  report-group-name-or-arn:
      files:
      - location
```

```
      - location
      base-directory: location
      discard-paths: no | yes
      file-format: report-format
artifacts:
  files:
      - location
      - location
  name: artifact-name
  discard-paths: no | yes
  base-directory: location
  exclude-paths: excluded paths
  enable-symlinks: no | yes
  s3-prefix: prefix
  secondary-artifacts:
      artifactIdentifier:
      files:
      - location
      - location
      name: secondary-artifact-name
      discard-paths: no | yes
      base-directory: location
      artifactIdentifier:
      files:
      - location
      - location
      discard-paths: no | yes
      base-directory: location
cache:
  paths:
      - path
      - path
```

Example:

```
version: 0.2

env:
  variables:
      JAVA_HOME: "/usr/lib/jvm/java-8-openjdk-amd64"
  parameter-store:
```

```
      LOGIN_PASSWORD: /CodeBuild/dockerLoginPassword

phases:
  install:
      commands:
      - echo Entered the install phase...
      - apt-get update -y
      - apt-get install -y maven
      finally:
      - echo This always runs even if the update or install command fails
  pre_build:
      commands:
      - echo Entered the pre_build phase...
      - docker login -u User -p $LOGIN_PASSWORD
      finally:
      - echo This always runs even if the login command fails
  build:
      commands:
      - echo Entered the build phase...
      - echo Build started on `date`
      - mvn install
      finally:
      - echo This always runs even if the install command fails
  post_build:
      commands:
      - echo Entered the post_build phase...
      - echo Build completed on `date`

reports:

arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group
-name-1:
      files:
      - "**/*"
      base-directory: 'target/tests/reports'
      discard-paths: no
  reportGroupCucumberJson:
      files:
      - 'cucumber/target/cucumber-tests.xml'
      discard-paths: yes
      file-format: CUCUMBERJSON # default is JUNITXML
artifacts:
  files:
```

```
      - target/messageUtil-1.0.jar
  discard-paths: yes
  secondary-artifacts:
      artifact1:
      files:
      - target/artifact-1.0.jar
      discard-paths: yes
      artifact2:
      files:
      - target/artifact-2.0.jar
      discard-paths: yes
cache:
  paths:
      - '/root/.m2/**/*'
```

Docker Sample

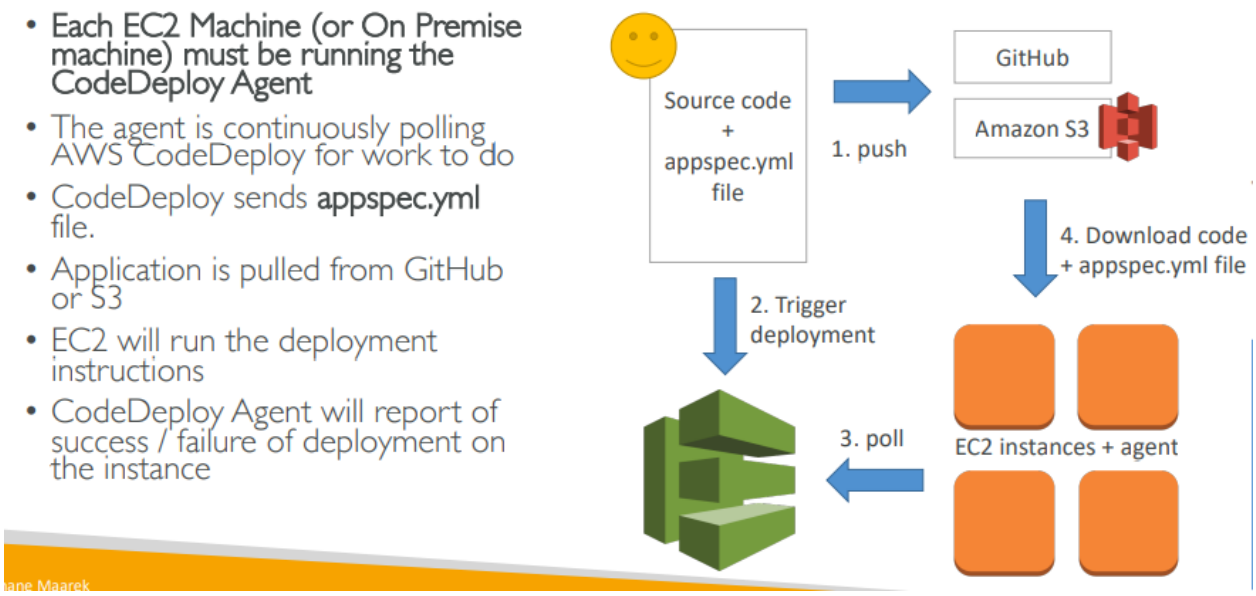
```
version: 0.2

phases:
  pre_build:
      commands:
      - echo Logging in to Amazon ECR...
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker
login --username AWS --password-stdin
$AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com
  build:
      commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
      - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG
$AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:
$IMAGE_TAG
  post_build:
      commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push
$AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:
```

```
$IMAGE_TAG
```

CodeDeploy

AWS CodeDeploy
• We want to deploy our application automatically to many EC2 instances
• There are several ways to handle deployments using open source tools (Ansible, Terraform, Chef, Puppet, etc…)
• We can use the managed Service AWS CodeDeploy



• EC2 instances are grouped by deployment group (dev / test / prod)
• Lots of flexibility to define any kind of deployments
• CodeDeploy can be chained into CodePipeline and use artifacts from there
• CodeDeploy can re-use existing setup tools, works with any application, auto scaling integration
• Note: Blue / Green only works with EC2 instances (not on premise)
• Support for AWS Lambda deployments, EC2
• CodeDeploy does not provision resources

EC2 CodeDeploy Configuration

```
sudo yum update -y
sudo yum install -y ruby wget
wget https://aws-codedeploy-eu-west-1.s3.eu-west-1.amazonaws.com/latest/install
```

```
chmod +x ./install
sudo ./install auto
sudo service codedeploy-agent status
```

Deployment Types

Blue Green Deployment

Replaces the instances in the deployment group with new instances and deploys the latest application revision to them. After instances in the replacement environment are registered with a load balancer, instances from the original environment are deregistered and can be terminated.

Environment Configuration(Blue Green Deployment)

- ☐ Automatically copy Amazon EC2 Auto Scaling group
  Provision an Amazon EC2 Auto Scaling group and deploy the new application revision to it. AWS CodeDeploy will create the Auto Scaling group by copying the one you specify here.

- ☐ Manually provision instances
  I will specify here the instances where the current application revision is running. I will specify the instances for the replacement environment when I create a deployment.

ELB presence is mandatory

In-place

Updates the instances in the deployment group with the latest application revisions. During a deployment, each instance will be briefly taken offline for its update
Events:
ApplicationStop → DownloadBundle → BeforeInstall → Install → AfterInstall → ApplicationStart → ValidateService

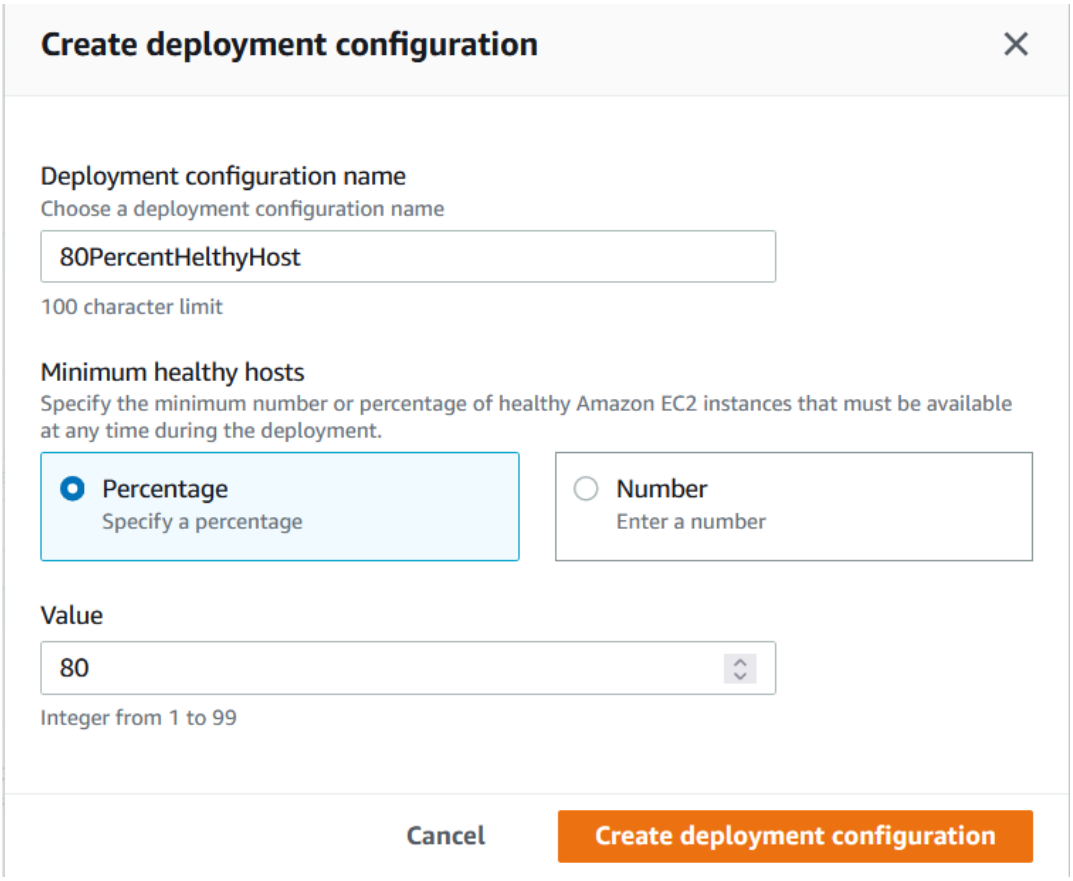
Deployment Settings/Configurations(For in-place)

A deployment configuration is a set of rules that determines how fast an application is deployed and the success or failure conditions for a deployment

Default Options:

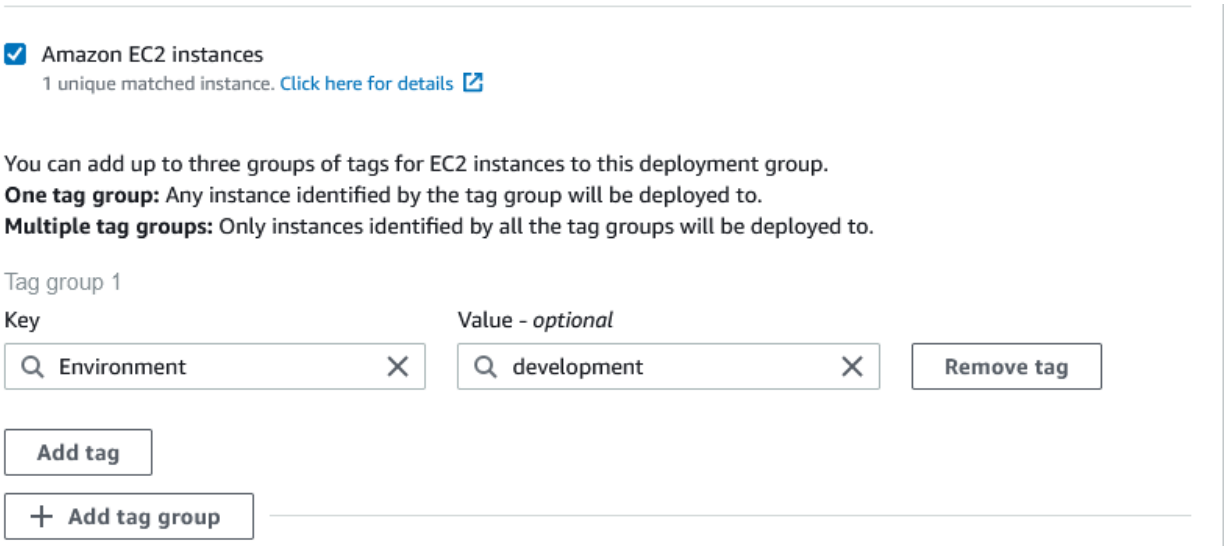
- AllAtOnce
- OneAtATime
- HalfAtTime

Custom Settings:



Environment configuration

- Tags to drive the deployments in EC2 instances.
- Tags to drive the creation of Deployment Groups.



appspec.yml

```yaml
version: 0.0
os: linux
files:
  - source: /index.html
    destination: /var/www/html/
hooks:
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root

  AfterInstall:
    - location: scripts/after_install.sh
      timeout: 300
      runas: root

  BeforeInstall:
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
```

```
ApplicationStart:
  - location: scripts/start_server.sh
    timeout: 300
    runas: root

ValidateService:
  - location: scripts/validate_service.sh
    timeout: 300
```

Hooks

https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file-structure-hooks.html

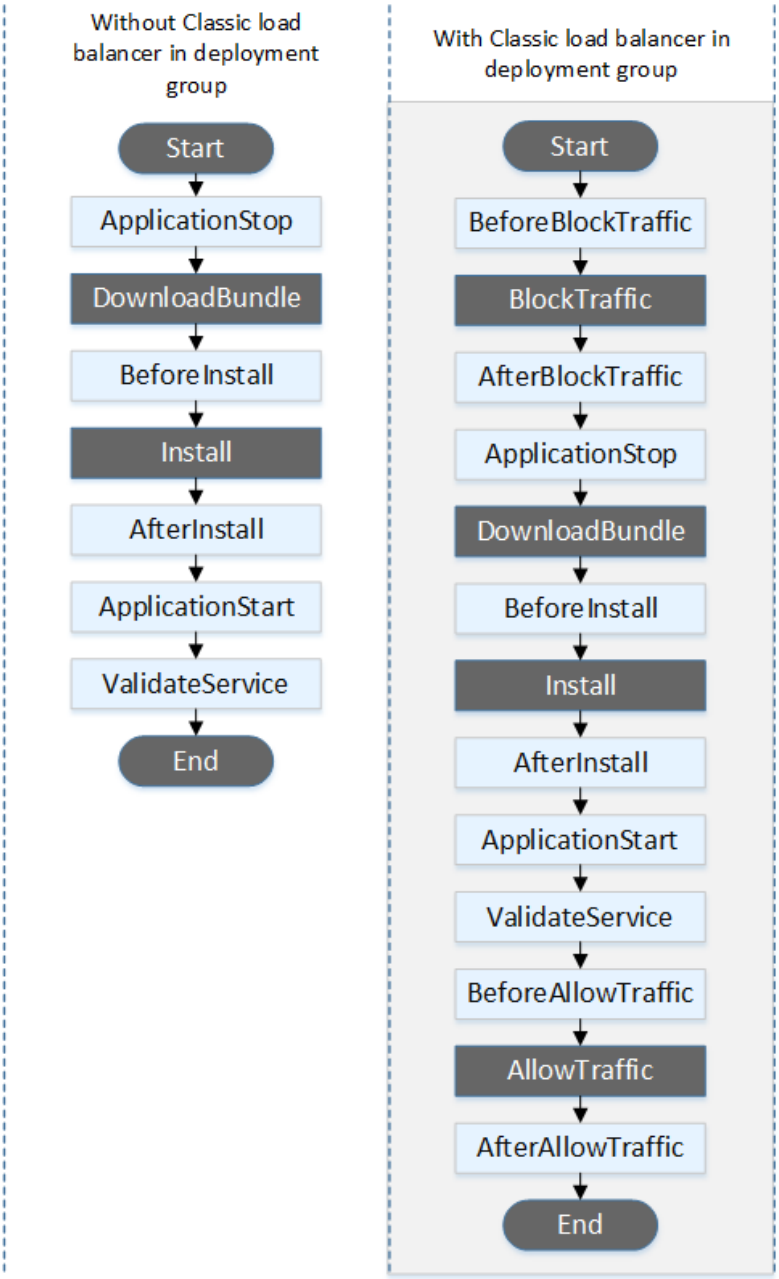
Lifecycle event name	In-place deployment[1]	Blue/green deployment: Original instances	Blue/green deployment: Replacement instances	Blue/green deployment rollback: Original instances	Blue/green deployment rollback: Replacement instances
ApplicationStop	✓		✓		
DownloadBundle[2]	✓		✓		
BeforeInstall	✓		✓		
Install[2]	✓		✓		
AfterInstall	✓		✓		
ApplicationStart	✓		✓		
ValidateService	✓		✓		
BeforeBlockTraffic	✓	✓			✓
BlockTraffic[2]	✓	✓			✓
AfterBlockTraffic	✓	✓			✓
BeforeAllowTraffic	✓		✓	✓	
AllowTraffic[2]	✓		✓	✓	
AfterAllowTraffic	✓		✓	✓	

Run order of hooks in a deployment

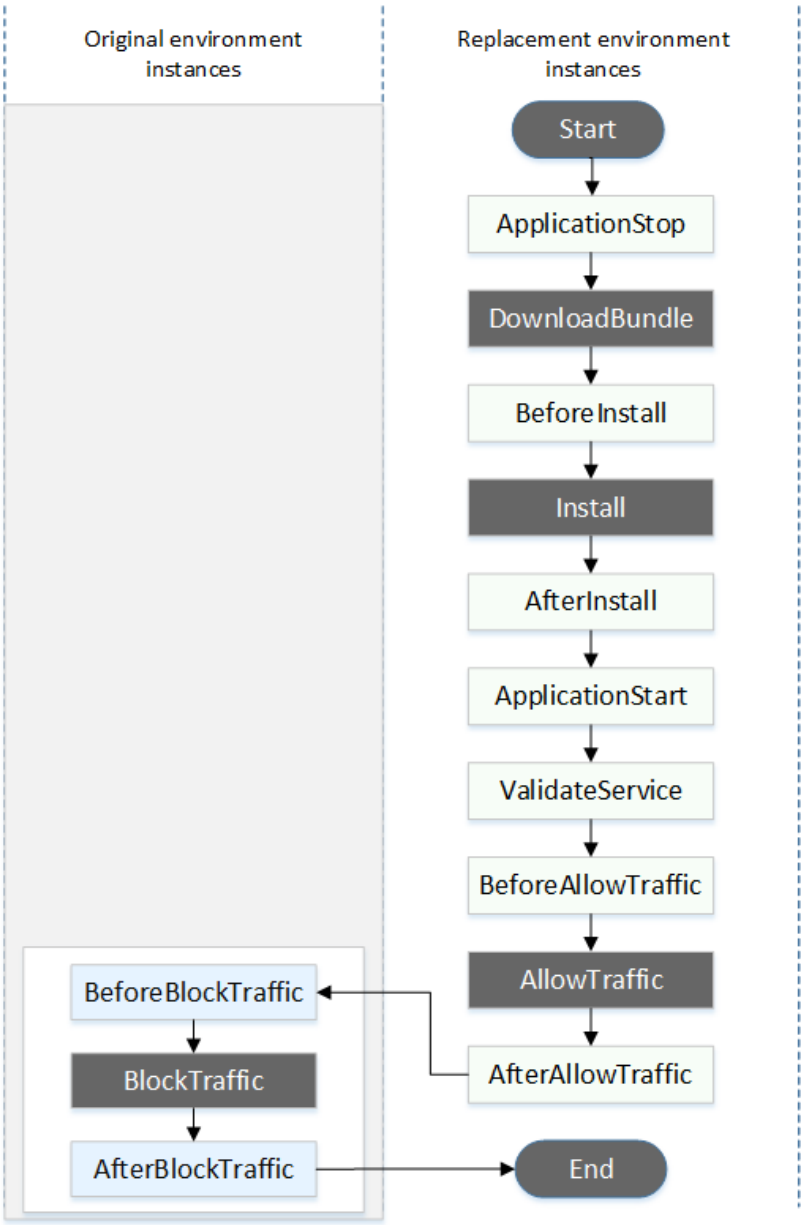
EC2/On-Premises deployment

In-place deployments

In an in-place deployment, including the rollback of an in-place deployment, event hooks are run in the following order



Blue Green Deployments



The **Start**, **DownloadBundle**, **Install**, **BlockTraffic**, **AllowTraffic**, and **End** events in the deployment cannot be scripted, which is why they appear in gray in this diagram. However, you can edit the 'files' section of the AppSpec file to specify what's installed during the **Install** event.

Environment variable availability for hooks

During each deployment lifecycle event, hook scripts can access the following environment variables:

APPLICATION_NAME

The name of the application in CodeDeploy that is part of the current deployment (for example, WordPress_App).

DEPLOYMENT_ID

The ID CodeDeploy has assigned to the current deployment (for example, d-AB1CDEF23).

DEPLOYMENT_GROUP_NAME

The name of the deployment group in CodeDeploy that is part of the current deployment (for example, WordPress_DepGroup).

DEPLOYMENT_GROUP_ID

The ID of the deployment group in CodeDeploy that is part of the current deployment (for example, b1a2189b-dd90-4ef5-8f40-4c1c5EXAMPLE).

LIFECYCLE_EVENT

The name of the current deployment lifecycle event (for example, AfterInstall).

These environment variables are local to each deployment lifecycle event.

The following script changes the listening port on an Apache HTTP server to 9090 instead of 80 if the value of **DEPLOYMENT_GROUP_NAME** is equal to Staging. This script must be invoked during the BeforeInstall deployment lifecycle event:

```
if [ "$DEPLOYMENT_GROUP_NAME" == "Staging" ]
then
    sed -i -e 's/Listen 80/Listen 9090/g' /etc/httpd/conf/httpd.conf
fi
```

The following script example changes the verbosity level of messages recorded in its error log from warning to debug if the value of the **DEPLOYMENT_GROUP_NAME** environment variable is equal to Staging. This script must be invoked during the BeforeInstall deployment lifecycle event:

```
if [ "$DEPLOYMENT_GROUP_NAME" == "Staging" ]
then
    sed -i -e 's/LogLevel warn/LogLevel debug/g' /etc/httpd/conf/httpd.conf
fi
```

Hooks Structure

```
hooks:
  deployment-lifecycle-event-name:
```

```
- location: script-location
  timeout: timeout-in-seconds
  runas: user-name
```

Monitoring deployments with Amazon CloudWatch Events

https://docs.aws.amazon.com/codedeploy/latest/userguide/monitoring-cloudwatch-events.html

Following types of targets when using CloudWatch Events as part of your CodeDeploy operations:

- AWS Lambda functions
- Kinesis streams
- Amazon SQS queues
- Built-in targets (EC2 CreateSnapshot API call, EC2 RebootInstances API call, EC2 StopInstances API call , and EC2 TerminateInstances API call)
- Amazon SNS topics

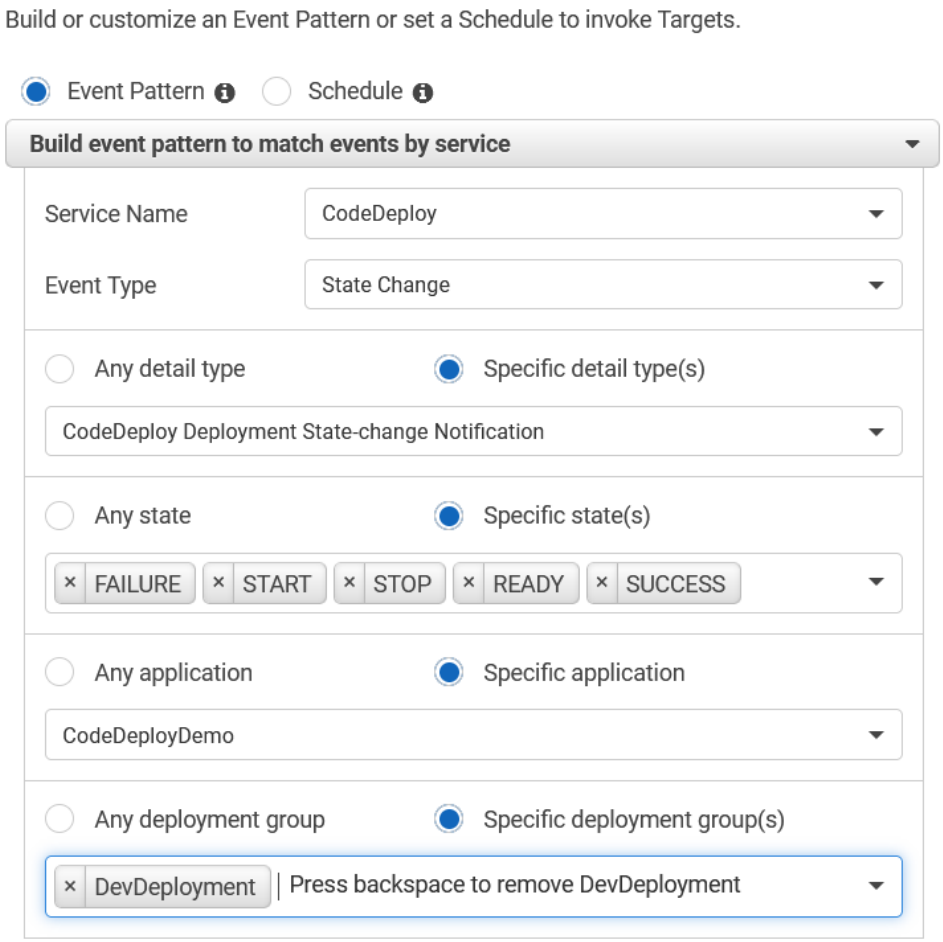
use cases:

- Use a Lambda function to pass a notification to a Slack channel whenever deployments fail.
- Push data about deployments or instances to a Kinesis stream to support comprehensive, real-time status monitoring.
- Use CloudWatch alarm actions to automatically stop, terminate, reboot, or recover Amazon EC2 instances when a deployment or instance event you specify occurs.

To make a rule that applies to some state changes only, choose **Specific state(s)**, and then choose one or more status values from the list. The following table lists the status values you can choose:

Deployment status values	Instance status values

FAILURE	FAILURE
START	START
STOP	READY
QUEUED	SUCCESS
READY	
SUCCESS	



Enable Cloudwatch Logs (AWS CodeDeploy)
Install the CloudWatch Logs agent

For new instances

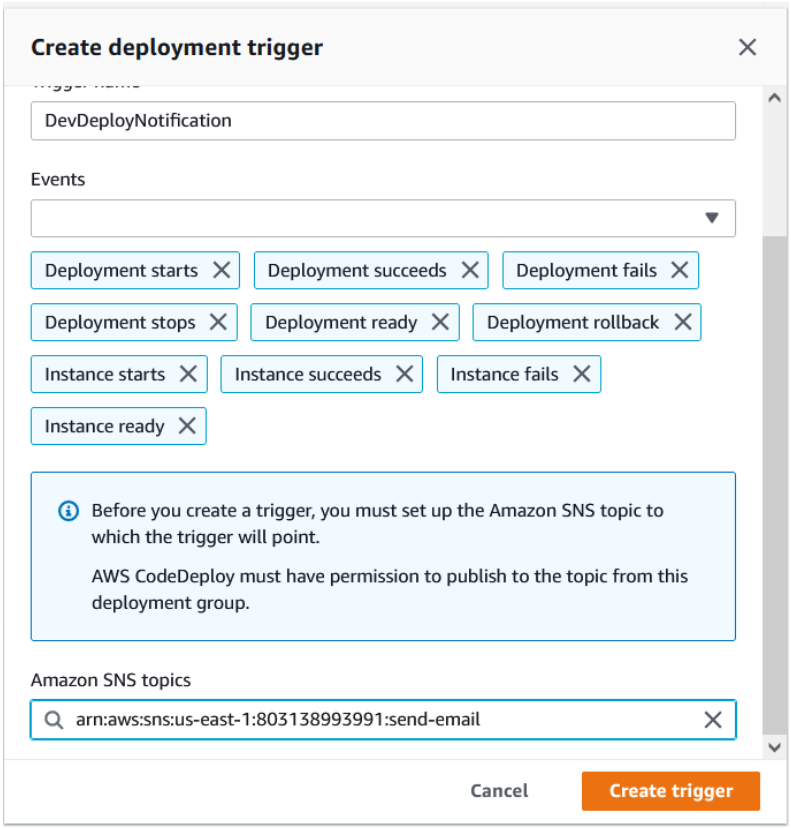
- Go to the EC2 console, and choose **Launch Instance**.
- Choose the desired AMI.
- Select the desired instance type, and then choose **Next**.

- On the **Configure Instance Details** page, for **IAM role**, choose the CodeDeploy deployment instance role.
- In **Advanced Details**, paste the following script in the **User data** area, and then choose **Review and Launch**.

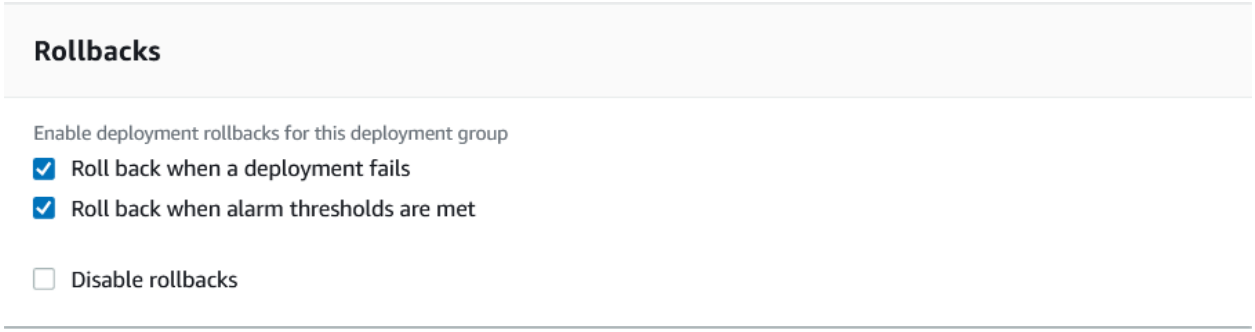
Substitute REGION in this script with the appropriate AWS region. For e.g. us-east-1 for US East (N. Virginia) region.

```
#!/bin/bash
wget https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-setup.py
wget https://s3.amazonaws.com/aws-codedeploy-us-east-1/cloudwatch/codedeploy_logs.conf
chmod +x ./awslogs-agent-setup.py
python awslogs-agent-setup.py -n -r REGION -c
s3://aws-codedeploy-us-east-1/cloudwatch/awslogs.conf
mkdir -p /var/awslogs/etc/config
cp codedeploy_logs.conf /var/awslogs/etc/config/
service awslogs restart
```

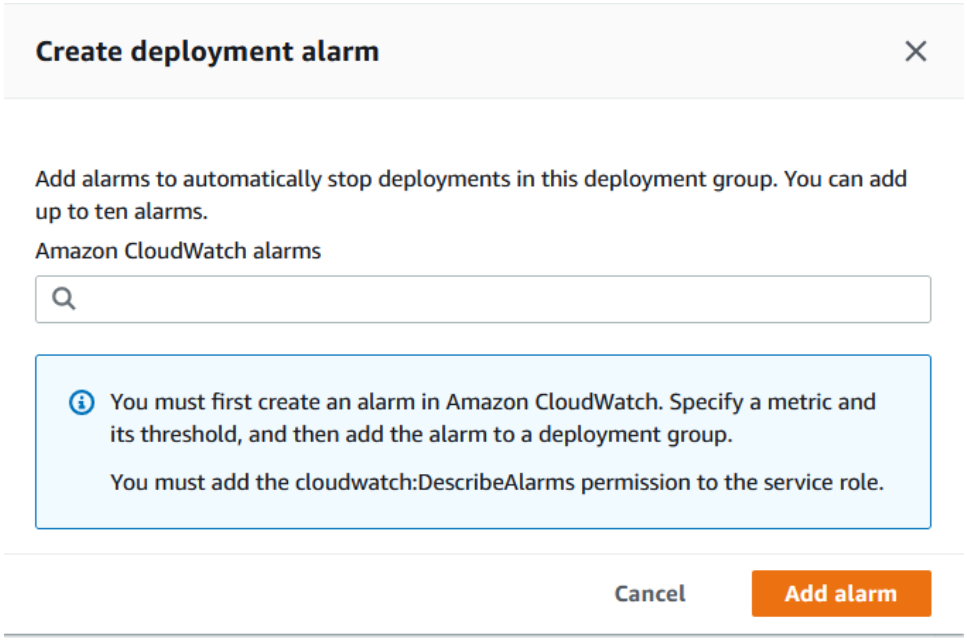
CodeDeploy Triggers (SNS Only)



Rollback



Amazon CloudWatch alarms: You can create a CloudWatch alarm that watches a single metric over a time period you specify and performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods. For an Amazon EC2 deployment, you can create an alarm for an instance or Amazon EC2 Auto Scaling group that you are using in your CodeDeploy operations. For an AWS Lambda and an Amazon ECS deployment, you can create an alarm for errors in a Lambda function.



You can configure a deployment to stop when an Amazon CloudWatch alarm detects that a metric has fallen below or exceeded a defined threshold.

You must have already created the alarm in CloudWatch before you can add it to **a deployment group**.

1. To add alarm monitoring to the deployment group, in **Alarms**, choose **Add alarm**.

2. Enter the name of a CloudWatch alarm you have already set up to monitor this deployment.
   You must enter the CloudWatch alarm exactly as it was created in CloudWatch. To view a list of alarms, open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/ and then choose **ALARM**.

Additional options:

- If you want deployments to proceed without taking into account alarms you have added, choose **Ignore alarm configuration**.
  This choice is useful when you want to temporarily deactivate alarm monitoring for a deployment group without having to add the same alarms again later.
- (Optional) If you want deployments to proceed in the event that CodeDeploy is unable to retrieve alarm status from Amazon CloudWatch, choose **Continue deployments even if alarm status is unavailable**.
- Note
  This option corresponds to **ignorePollAlarmFailure** in the AlarmConfiguration object in the CodeDeploy API.

For more information, see Monitoring deployments with CloudWatch alarms in CodeDeploy.

Automatic rollbacks: You can configure a deployment group or deployment to automatically roll back when a deployment fails or when a monitoring threshold you specify is met. In this case, the last known good version of an application revision is deployed. You can configure optional settings for a deployment group when you use the console to create an application, create a deployment group, or update a deployment group. When you create a new deployment, you can also choose to override the automatic rollback configuration that were specified for the deployment group.

- You can enable deployments to roll back to the most recent known good revision when something goes wrong by choosing one or both of the following:
  - **Roll back when a deployment fails**. CodeDeploy will redeploy the last known good revision as a new deployment.
  - **Roll back when alarm thresholds are met**. If you added an alarm to this application in the previous step, CodeDeploy will redeploy the last known good revision when one or more of the specified alarms is activated.

- Note
  To temporarily ignore a rollback configuration, choose **Disable rollbacks**. This choice is useful when you want to temporarily disable automatic rollbacks without having to set up the same configuration again later.
  For more information, see Redeploy and roll back a deployment with CodeDeploy.

Automatic updates to outdated instances: Under certain circumstances, CodeDeploy may deploy an outdated revision of your application to your Amazon EC2 instances. For example, if your EC2 instances are launched into an Auto Scaling group (ASG) while a CodeDeploy deployment is underway, those instances receive the older revision of your application instead of the latest one. To bring those instances up to date, CodeDeploy automatically starts a follow-on deployment (immediatedly after the first) to update any outdated instances. If you'd like to change this default behavior so that outdated EC2 instances are left at the older revision, you can do so through the CodeDeploy API or the AWS Command Line Interface (CLI).

To configure automatic updates of outdated instances through the API, include the outdatedInstancesStrategy request parameter in the UpdateDeploymentGroup or CreateDeploymentGroup action. For details, see the *AWS CodeDeploy API Reference*.

To configure the automatic updates through the AWS CLI, use one of the following commands:

aws deploy update-deployment-group arguments --outdated-instances-strategy UPDATE|IGNORE

Or...

aws deploy create-deployment-group arguments --outdated-instances-strategy UPDATE|IGNORE

...where arguments is replaced with the arguments required for your deployment, and UPDATE|IGNORE is replaced with either UPDATE to enable auto-updates, or IGNORE to disable them.

Example:

aws deploy update-deployment-group --application-name "MyApp" --current-deployment-group-name "MyDG" --region us-east-1 --outdated-instances-strategy IGNORE

Working with on-premises instances for CodeDeploy

An on-premises instance is any physical device that is not an Amazon EC2 instance that can run the CodeDeploy agent and connect to public AWS service endpoints.

Deploying a CodeDeploy application revision to an on-premises instance involves two major steps:

- **Step 1** – Configure each on-premises instance, register it with CodeDeploy, and then tag it.
- **Step 2** – Deploy application revisions to the on-premises instance.

If you don't want an on-premises instance to be used in deployments anymore, you can remove the on-premises instance tags from the deployment groups. For a more robust approach, remove the on-premises instance tags from the instance. You can also explicitly deregister an on-premises instance so it can no longer be used in any deployments.

Prerequisites:

- The device you want to prepare, register, and tag as an on-premises instance with CodeDeploy must be running a supported operating system.
- The on-premises instance must be able to connect to public AWS service endpoints to communicate with CodeDeploy
- The local or network account used on the on-premises instance to configure the on-premises instance must be able to run either as sudo or root
- The IAM identity you use to register the on-premises instance must be granted permissions to complete the registration (and to deregister the on-premises instance, as needed).

Additional permissions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateAccessKey",
        "iam:CreateUser",
        "iam:DeleteAccessKey",
        "iam:DeleteUser",
        "iam:DeleteUserPolicy",
        "iam:ListAccessKeys",
        "iam:ListUserPolicies",
        "iam:PutUserPolicy",
        "iam:GetUser"
      ],
      "Resource": "*"
    }
  ]
}
```

Deployment configurations on an AWS Lambda compute platform

When you deploy to an AWS Lambda compute platform, the deployment configuration specifies the way traffic is shifted to the new Lambda function versions in your application.

There are three ways traffic can shift during a deployment:

- **Canary**: Traffic is shifted in two increments. You can choose from predefined canary options that specify the percentage of traffic shifted to your updated Lambda function version in the first increment and the interval, in minutes, before the remaining traffic is shifted in the second increment.

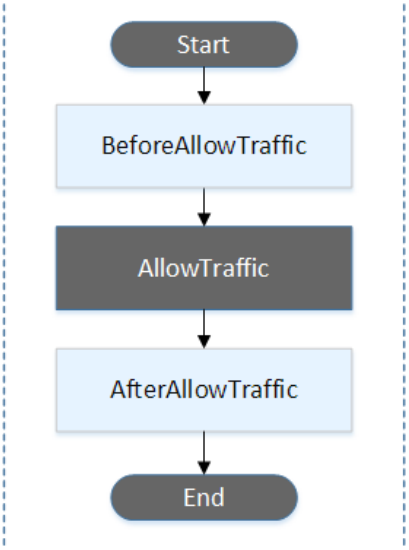
- **Linear**: Traffic is shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic shifted in each increment and the number of minutes between each increment.

- **All-at-once**: All traffic is shifted from the original Lambda function to the updated Lambda function version all at once.

An AWS Lambda hook is one Lambda function specified with a string on a new line after the name of the lifecycle event. Each hook is executed once per deployment. Here are descriptions of the hooks available for use in your AppSpec file.

- **BeforeAllowTraffic** – Use to run tasks before traffic is shifted to the deployed Lambda function version.

- **AfterAllowTraffic** – Use to run tasks after all traffic is shifted to the deployed Lambda function version.



Use the 'hooks' section to specify a Lambda function that CodeDeploy can call to validate a Lambda deployment. You can use the same function or a different one for the BeforeAllowTraffic and AfterAllowTraffic deployment lifecyle events.

Before invoking a Lambda hook function, the server must be notified of the deployment ID and the lifecycle event hook execution ID using the putLifecycleEventHookExecutionStatus command.

The following is a sample Lambda hook function written in Node.js.

```
'use strict';

const aws = require('aws-sdk');
const codedeploy = new aws.CodeDeploy({apiVersion: '2014-10-06'});

exports.handler = (event, context, callback) => {
    //Read the DeploymentId from the event payload.
    var deploymentId = event.DeploymentId;

    //Read the LifecycleEventHookExecutionId from the event payload
    var lifecycleEventHookExecutionId =
event.LifecycleEventHookExecutionId;

    /*
     Enter validation tests here.
    */

    // Prepare the validation test results with the deploymentId and
    // the lifecycleEventHookExecutionId for CodeDeploy.
    var params = {
        deploymentId: deploymentId,
        lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
        status: 'Succeeded' // status can be 'Succeeded' or 'Failed'
    };

    // Pass CodeDeploy the prepared validation test results.
    codedeploy.putLifecycleEventHookExecutionStatus(params, function(err,
data) {
        if (err) {
            // Validation failed.
            callback('Validation test failed');
        } else {
            // Validation succeeded.
```

```
            callback(null, 'Validation test succeeded');
        }
    });
};
```

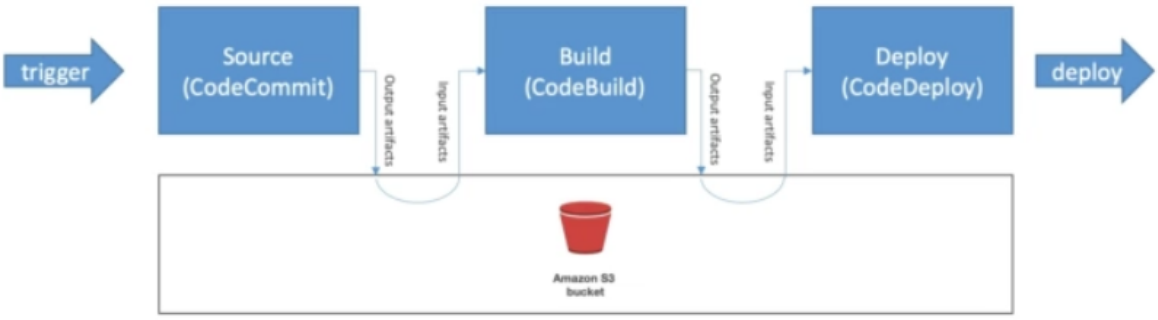
CodePipeline

CodePipeline

- Continuous delivery
- Visual workflow
- Source: GitHub / CodeCommit / Amazon S3
- Build: CodeBuild / Jenkins / etc…
- Load Testing: 3rd party tools
- Deploy: AWS CodeDeploy / Beanstalk / CloudFormation / ECS…
- Made of stages:
  - Each stage can have sequential actions and / or parallel actions
  - Stages examples: Build / Test / Deploy / Load Test / etc…
  - Manual approval can be defined at any stage

AWS CodePipeline Artifacts

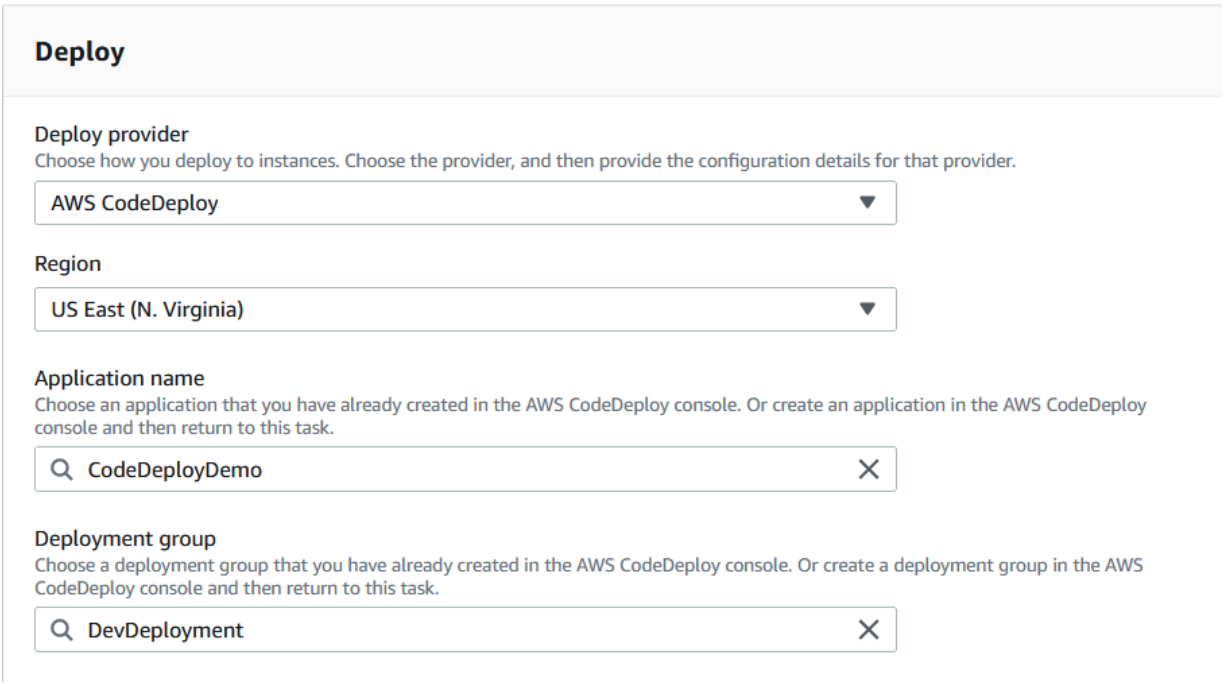
- Each pipeline stage can create "artifacts"
- Artifacts are passed stored in Amazon S3 and passed on to the next stage



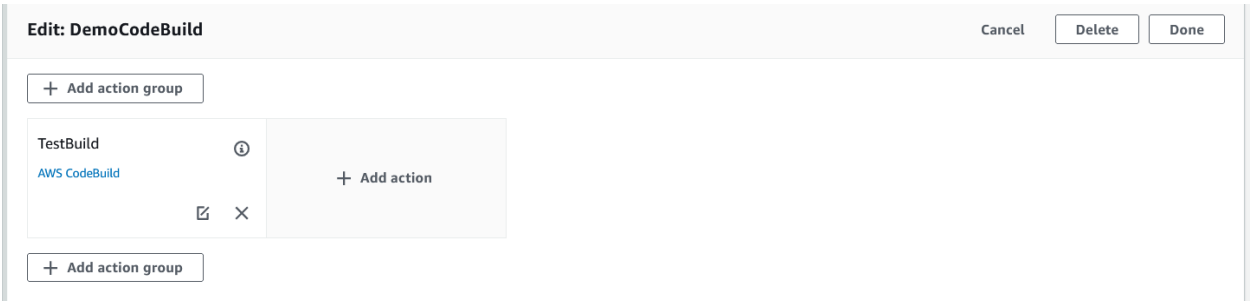
Artifact store
Default location
Use the default artifact store (Amazon S3 codepipeline-us-east-1-119434769434) designated in the same region and account as your pipeline
Custom location
Choose an existing S3 location from your account in the same region and account as your pipeline

Output artifact format
CodePipeline default
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include git metadata about the repository.
Full clone
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full git clone. Only supported for AWS CodeBuild actions.

Change detection options
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.
Amazon CloudWatch Events (recommended)
Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs
AWS CodePipeline
Use AWS CodePipeline to check periodically for changes



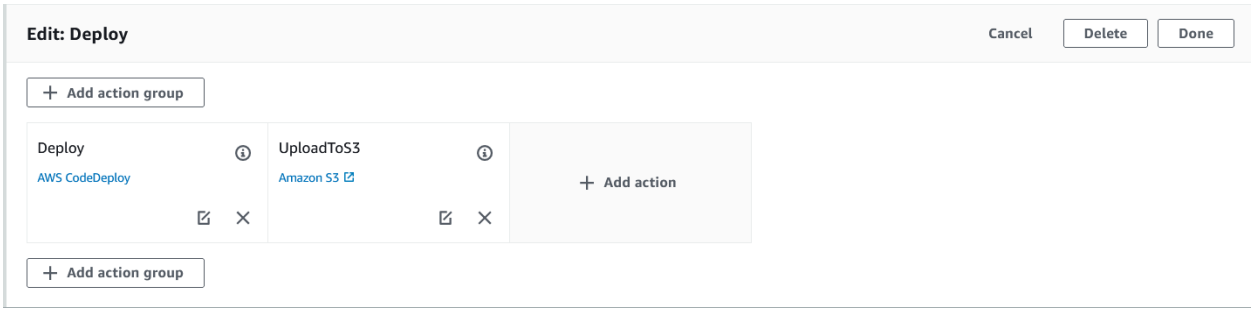
AWS CodePipeline can trigger AWS CodeDeploy/CodeBuild service, created in another region.



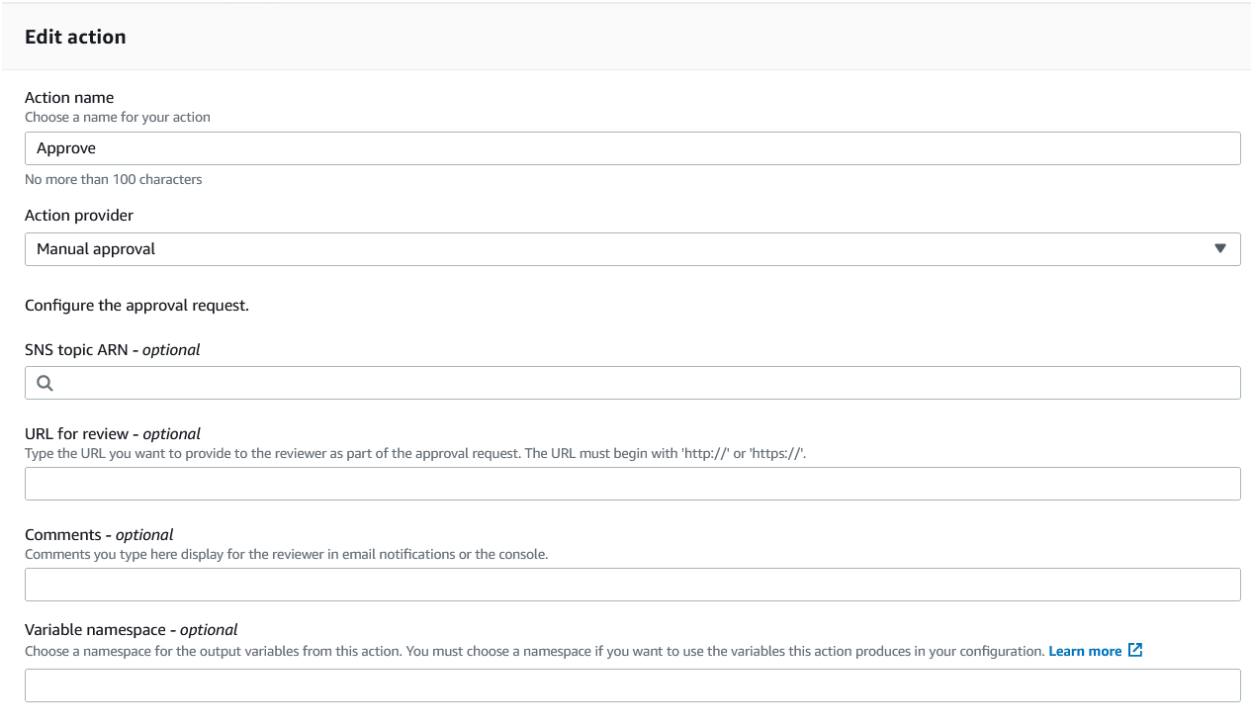
CodePipeline execution steps

a. Deployment Stages → sequential
   - Action Groups → sequential
     1. Actions → parallel

\+ Add Action Group → sequential actions
\+ Action → parallel actions



Approval:



Cloudwatch Events

- CodePipeline automatically creates a CodeCommit cloudwatch event(which triggers codepipeline deployment on code commit.)
- CodePipeline CloudWatch events:
    1. CodePipeline Pipeline Execution State Change
    2. CodePipeline Stage Execution State Change
    3. CodePipeline Action Execution State Change

```
{
  "source": [
      "aws.codepipeline"
  ],
  "detail-type": [
```

```
      "CodePipeline Action Execution State Change"
  ],
  "detail": {
      "state": [
      "FAILED",
      "SUCCEEDED",
      "CANCELED",
      "STARTED"
      ]
  }
}
```

CodePipeline As Code:
https://docs.aws.amazon.com/codepipeline/latest/userguide/reference-pipeline-structure.html

- runOrder → to define order of execution(same runOrder → parallel execution)

https://docs.aws.amazon.com/codepipeline/latest/userguide/best-practices.html

Invoke an AWS Lambda function in a pipeline in CodePipeline

Ways Lambda functions can be used in pipelines:

- To create resources on demand in one stage of a pipeline using AWS CloudFormation and delete them in another stage.
- To deploy application versions with zero downtime in AWS Elastic Beanstalk with a Lambda function that swaps CNAME values.
- To deploy to Amazon ECS Docker instances.
- To back up resources before building or deploying by creating an AMI snapshot.
- To add integration with third-party products to your pipeline, such as posting messages to an IRC client.

Sample CodePipeline json:
aws codepipeline get-pipeline --name CodePipeLineDemo

```
{
      "pipeline": {
      "name": "CodePipeLineDemo",
      "roleArn":
"arn:aws:iam::803138993991:role/service-role/AWSCodePipelineServiceRole-us-
east-1-CodePipeLineDemo",
      "artifactStore": {
```

```
            "type": "S3",
            "location": "aws-devops-course-samriddha"
      },
      "stages": [
            {
                  "name": "Source",
                  "actions": [
                  {
                        "name": "Source",
                        "actionTypeId": {
                              "category": "Source",
                              "owner": "AWS",
                              "provider": "CodeStarSourceConnection",
                              "version": "1"
                        },
                        "runOrder": 1,
                        "configuration": {
                              "BranchName": "main",
                              "ConnectionArn":
"arn:aws:codestar-connections:us-east-1:803138993991:connection/91634fb8-d3
e1-4efb-9873-a7240c0b166b",
                              "FullRepositoryId":
"samriddhac/demo-aws-web-server",
                              "OutputArtifactFormat": "CODE_ZIP"
                        },
                        "outputArtifacts": [
                              {
                              "name": "SourceArtifact"
                              }
                        ],
                        "inputArtifacts": [],
                        "region": "us-east-1",
                        "namespace": "SourceVariables"
                  }
                  ]
            },
            {
                  "name": "DemoCodeBuild",
                  "actions": [
                  {
                        "name": "TestBuild",
                        "actionTypeId": {
                              "category": "Test",
```

```
                        "owner": "AWS",
                        "provider": "CodeBuild",
                        "version": "1"
                    },
                    "runOrder": 1,
                    "configuration": {
                        "ProjectName": "CodeBuildDemo"
                    },
                    "outputArtifacts": [
                        {
                        "name": "Results"
                        }
                    ],
                    "inputArtifacts": [
                        {
                        "name": "SourceArtifact"
                        }
                    ],
                    "region": "us-east-1",
                    "namespace": "Test"
                }
                ]
            },
            {
                "name": "Approval",
                "actions": [
                {
                    "name": "Approve",
                    "actionTypeId": {
                        "category": "Approval",
                        "owner": "AWS",
                        "provider": "Manual",
                        "version": "1"
                    },
                    "runOrder": 1,
                    "configuration": {},
                    "outputArtifacts": [],
                    "inputArtifacts": [],
                    "region": "us-east-1"
                }
                ]
            },
            {
```

```
                "name": "Deploy",
                "actions": [
                {
                      "name": "Deploy",
                      "actionTypeId": {
                            "category": "Deploy",
                            "owner": "AWS",
                            "provider": "CodeDeploy",
                            "version": "1"
                      },
                      "runOrder": 1,
                      "configuration": {
                            "ApplicationName": "CodeDeployDemo",
                            "DeploymentGroupName": "DevDeployment"
                      },
                      "outputArtifacts": [],
                      "inputArtifacts": [
                            {
                            "name": "SourceArtifact"
                            }
                      ],
                      "region": "us-east-1",
                      "namespace": "DeployVariables"
                },
                {
                      "name": "UploadToS3",
                      "actionTypeId": {
                            "category": "Deploy",
                            "owner": "AWS",
                            "provider": "S3",
                            "version": "1"
                      },
                      "runOrder": 1,
                      "configuration": {
                            "BucketName": "aws-devops-course-samriddha",
                            "Extract": "false",
                            "ObjectKey": "demo-code-build-results"
                      },
                      "outputArtifacts": [],
                      "inputArtifacts": [
                            {
                            "name": "SourceArtifact"
                            }
```

```
                        ],
                        "region": "us-east-1"
                  },
                  {
                        "name": "Verification",
                        "actionTypeId": {
                              "category": "Invoke",
                              "owner": "AWS",
                              "provider": "Lambda",
                              "version": "1"
                        },
                        "runOrder": 2,
                        "configuration": {
                              "FunctionName":
"MyLambdaFunctionForAWSCodePipeline",
                              "UserParameters":
"http://ec2-54-89-63-132.compute-1.amazonaws.com"
                        },
                        "outputArtifacts": [],
                        "inputArtifacts": [
                              {
                              "name": "SourceArtifact"
                              }
                        ],
                        "region": "us-east-1"
                  }
                  ]
            }
      ],
      "version": 6
      },
      "metadata": {
      "pipelineArn":
"arn:aws:codepipeline:us-east-1:803138993991:CodePipeLineDemo",
      "created": "2021-10-03T12:38:52.143000+05:30",
      "updated": "2021-10-03T19:29:17.737000+05:30"
      }
}
```

https://docs.aws.amazon.com/codepipeline/latest/userguide/tutorials-cloudformation.html

Jenkins

Jenkins on AWS

- Open Source CICD tool
- Can replace CodeBuild, CodePipeline & CodeDeploy
- Must be deployed in a Master / Slave configuration
- Must manage multi-AZ, deploy on EC2, etc...
- All projects must have a "Jenkinsfile" (similar to buildspec.yml) to tell Jenkins what to do

- Jenkins can be extended on AWS thanks to many plugins!

Jenkins Master / Slave (build farm)

Figure 2: Master and Worker deployment options

Jenkins on AWS

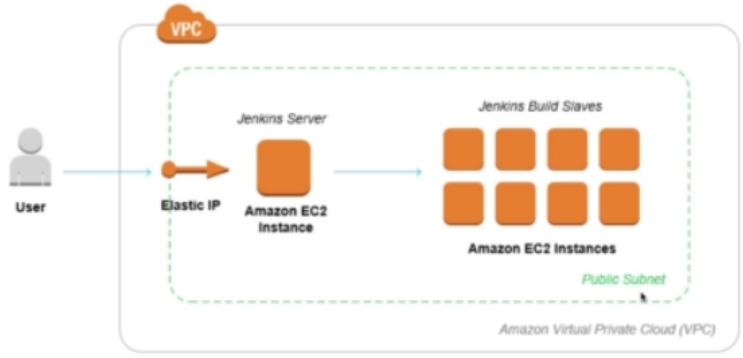


Figure 3: Jenkins deployment strategies

Jenkins with CodePipeline

Figure 9: A simple workflow using AWS services and Jenkins

Jenkins with ECS

Figure 10: Using AWS services and Jenkins to deploy a container

Jenkins with Device Farm

Figure 11: Using AWS services and Jenkins to test a mobile application

Jenkins with AWS Lambda

Figure 12: Using AWS and Jenkins for serverless code management

Jenkins with CloudFormation

Figure 13: First example—Using AWS services and Jenkins to automate a security framework

AWS Jenkins Plugins

- Ec2 plugin
- Codebuild plugin

TODO

https://aws.amazon.com/blogs/devops/validating-aws-codecommit-pull-requests-with-aws-codebuild-and-aws-lambda/
https://docs.aws.amazon.com/codedeploy/latest/userguide/instances-on-premises-register-instance.html
https://docs.aws.amazon.com/codedeploy/latest/userguide/register-on-premises-instance-iam-user-arn.html
https://docs.aws.amazon.com/codedeploy/latest/userguide/register-on-premises-instance-iam-session-arn.html
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments-create-ecs-cfn.html
https://aws.amazon.com/blogs/devops/implementing-gitflow-using-aws-codepipeline-aws-codecommit-aws-codebuild-and-aws-codedeploy/

BLOGS

- **MUST READ -** Blue/Green Deployments on AWS
  https://d1.awsstatic.com/whitepapers/AWS_Blue_Green_Deployments.pdf
- **RECOMMENDED -** Practicing Continuous Integration Continuous Delivery on AWS

  https://d1.awsstatic.com/whitepapers/DevOps/practicing-continuous-integration-continuous-delivery-on-AWS.pdf
- **RECOMMENDED -** Jenkins on AWS
  https://d1.awsstatic.com/whitepapers/DevOps/Jenkins_on_AWS.pdf
- **OPTIONAL -** Introduction to DevOps on AWS
  https://d1.awsstatic.com/whitepapers/AWS_DevOps.pdf
- **OPTIONAL -** Development and Test on AWS
  https://d1.awsstatic.com/whitepapers/aws-development-test-environments.pdf

GOOD LINKS:

https://github.com/aws-samples