St. Francis Institute of Technology, Mumbai-400 103

**Department Of Information Technology**

A.Y. 2021-2022

Class: TE-ITA/B, Semester: VIII

Subject: **DevOps Lab**

**Experiment – 11: Case study on version control and continuous integration tools** (Topic Beyond Syllabus)

1. **Aim:** To prepare case study on version control and continuous integration tools
2. **Objectives:** Aim of this experiment is that, the students will be able

* To be aware of different Version Control tools like Subversion, Bitbucket etc.
* To be aware of different continuous integration tools like Travis CI, Bamboo etc.

1. **Outcomes:** After study of this experiment, the students will be able to know about

* Version Control tools
* continuous integration tools

1. **Prerequisite:** Knowledge of software engineering concept of version control and integration
2. **Requirements:** Personal Computer, Windows operating system, browser, Internet Connection, Microsoft Word.
3. **Pre-Experiment Exercise:**

**Brief Theory:** Refer shared material

1. **Laboratory Exercise**
   * + 1. **Procedure:**

**a. Case study on version control:**

* Select any one version control tool like AWS codecommit, Mercurial, Subversion, Bitbucket, CVS etc.
* Prepare case study on any one of the tools covering following points:
  + Introduction
  + Features
  + Installation and Working
  + Advantages
  + Disadvantages

**b**. **Case study on continuous integration:**

* Select any one continuous integration tool like Travis CI, Bamboo, GitLab, AWS CodePipeline etc.
* Prepare case study on any one of the tools covering following points:
  + Introduction
  + Features
  + Installation and Working
  + Advantages
  + Disadvantages

1. **Post-Experiments Exercise**
2. **Extended Theory:**

Nil

1. **Questions:**

Nil

1. **Conclusion:**

* Write what was performed in the experiment.
* Write the significance of the topic studied in the experiment.

1. **References:** *(provide your own reference links)*

* [**https://aws.amazon.com/codecommit/**](https://aws.amazon.com/codecommit/)
* [**https://aws.amazon.com/codecommit/features/**](https://aws.amazon.com/codecommit/features/)
* [**https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up.html**](https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up.html)
* [**https://aws.amazon.com/codepipeline/**](https://aws.amazon.com/codepipeline/)
* [**https://aws.amazon.com/codepipeline/features/?nc=sn&loc=2**](https://aws.amazon.com/codepipeline/features/?nc=sn&loc=2)
  + [**https://docs.aws.amazon.com/codepipeline/latest/userguide/getting-started-codepipeline.html**](https://docs.aws.amazon.com/codepipeline/latest/userguide/getting-started-codepipeline.html)
  + [**https://abhinavmanc.medium.com/aws-codepipeline-advantages-and- disadvantages-5ceba8f4f3f3**](https://abhinavmanc.medium.com/aws-codepipeline-advantages-and-%20%20%20disadvantages-5ceba8f4f3f3)

**Laboratory exercise:**

**Case study on version control AWS CodeCommit:**

* **Introduction:**

AWS CodeCommit is a secure, highly scalable, managed [source control](https://aws.amazon.com/devops/source-control/) service that hosts private Git repositories. It makes it easy for teams to securely collaborate on code with contributions encrypted in transit and at rest. CodeCommit eliminates the need for you to manage your own source control system or worry about scaling its infrastructure. You can use CodeCommit to store anything from code to binaries. It supports the standard functionality of Git, so it works seamlessly with your existing Git-based tools.

* **Features**

AWS CodeCommit is a highly scalable, managed source control service that hosts private Git repositories. You simply create a repository to store your code. There is no hardware to provision and scale or software to install, configure, and operate. CodeCommit helps you collaborate on code with pull requests, branching, and merging. You can implement workflows that include code reviews and feedback by default, and control who can make changes to specific branches.

**Collaboration**

AWS CodeCommit is designed for collaborative software development. You can easily commit, branch, and merge your code allowing you to easily maintain control of your team’s projects. CodeCommit also supports pull requests, which provide a mechanism to request code reviews and discuss code with collaborators. You can create a repository from the AWS Management Console, AWS CLI, or AWS SDKs and start working with the repository using Git.

**Encryption**

You can transfer your files to and from AWS CodeCommit using HTTPS or SSH, as you prefer. Your repositories are also automatically encrypted at rest through AWS Key Management Service (AWS KMS) using customer-specific keys.

**Access Control**

AWS CodeCommit uses AWS Identity and Access Management to control and monitor who can access your data as well as how, when, and where they can access it. CodeCommit also helps you monitor your repositories via AWS CloudTrail and AWS CloudWatch.

**High Availability and Durability**

AWS CodeCommit stores your repositories in Amazon S3 and Amazon DynamoDB.  Your encrypted data is redundantly stored across multiple facilities. This architecture increases the availability and durability of your repository data.

**The repositories you need, when you need them**

AWS CodeCommit allows you to create up to 1,000 repositories by default, and additional repositories up to 25,000 by request. You can store and version any kind of file, including application assets such as images and libraries alongside your code. It’s easy to create repositories when you need them, and delete them when you’re done.

**Easy Access and Integration**

You can use the AWS Management Console, AWS CLI, and AWS SDKs to manage your repositories. You can also use Git commands or Git graphical tools to interact with your repository source files. AWS CodeCommit supports all Git commands and works with your existing Git tools. You can integrate with your development environment plugins or continuous integration/continuous delivery systems.

**Notifications and Custom Scripts**

You can now receive notifications for events impacting your repositories. Notifications will come in the form of Amazon SNS notifications. Each notification will include a status message as well as a link to the resources whose event generated that notification. Additionally, using AWS CodeCommit repository triggers, you can send notifications and create HTTP webhooks with Amazon SNS or invoke AWS Lambda functions in response to the repository events you choose.

* **Installation and Working**

Before you can connect to AWS CodeCommit for the first time, you must complete the initial configuration steps. For most users, this can be done most easily by following the steps in [For HTTPS users using Git credentials](https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-gc.html).

**Step1**: Initial configuration for CodeCommit

**Step2:** Install Git

**Step3:** Create Git credentials for HTTPS connections to CodeCommit

**Step4:** Connect to the CodeCommit console and clone the repository

However, if you want to connect to CodeCommit using a root account, federated access, or temporary credentials, you can use the credential helper that is included in the AWS CLI.

**Step1:** Initial configuration for CodeCommit

**Step2:** Install Git

**Step3:** Set up the credential helper

**Step4:** Connect to the CodeCommit console and clone the repository

Getting started with AWS CodeCommit, the following steps shows you how to use some key CodeCommit features. First, you create a repository and commit some changes to it. Then, you browse the files and view the changes. You can also create a pull request so other users can review and comment on changes to your code.

**Step1:** Create a CodeCommit repository

**Step2:** Add files to your repository

**Step3:** Browse the contents of your repository

**Step4:** Create and collaborate on a pull request

**Step5:** Clean up

If you are new to Git and CodeCommit, In the steps, you create a repository that represents a local copy of the CodeCommit repository, which we refer to as a *local repo*.After you create the local repo, you make some changes to it. Then you send (push) your changes to the CodeCommit repository.You also simulate a team environment where two users independently commit changes to their local repo and push those changes to the CodeCommit repository. The users then pull the changes from the CodeCommit repository to their own local repo to see the changes the other user made.You also create branches and tags and manage some access permissions in the CodeCommit repository.

**Step1:** Create a CodeCommit repository

**Step2:** Create a local repo

**Step3:** Create your first commit

**Step4:** Push your first commit

**Step5:** Share the CodeCommit repository and push and pull another commit

**Step6:** Create and share a branch

**Step7:** Create and share a tag

**Step8:** Set up access permissions

**Step9:** Clean up

* **Advantages**
* **Benefit from a fully managed service hosted by AWS**. CodeCommit provides high service availability and durability and eliminates the administrative overhead of managing your own hardware and software. There is no hardware to provision and scale and no server software to install, configure, and update.
* **Store your code securely.** CodeCommit repositories are encrypted at rest as well as in transit.
* **Work collaboratively on code.** CodeCommit repositories support pull requests, where users can review and comment on each other's code changes before merging them to branches; notifications that automatically send emails to users about pull requests and comments; and more.
* **Easily scale your version control projects.** CodeCommit repositories can scale up to meet your development needs. The service can handle repositories with large numbers of files or branches, large file sizes, and lengthy revision histories.
* **Store anything, anytime**. CodeCommit has no limit on the size of your repositories or on the file types you can store.
* **Integrate with other AWS and third-party services.** CodeCommit keeps your repositories close to your other production resources in the AWS Cloud, which helps increase the speed and frequency of your development lifecycle. It is integrated with IAM and can be used with other AWS services and in parallel with other repositories. For more information, see Product and service integrations with AWS CodeCommit.
* **Easily migrate files from other remote repositories**. You can migrate to CodeCommit from any Git-based repository.
* **Use the Git tools you already know.** CodeCommit supports Git commands as well as its own AWS CLI commands and APIs.
* **Disadvantages**
* complex menu options
* resubmit flow and code review unavailable
* limited triggers
* lack of CI system integrations

**Case Study on Continuous Integraion AWS CodePipeline**

* **Introduction**

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. There are no upfront fees or long-term commitments

* **Features**

AWS CodePipeline is a continuous integration and continuous delivery service for fast and reliable application and infrastructure updates. You can use CodePipeline to fully model and automate your software release processes.

**Workflow modeling**

A pipeline defines your release process workflow, and describes how a new code change progresses through your release process. A pipeline comprises a series of stages (e.g., build, test, and deploy), which act as logical divisions in your workflow. Each stage is made up of a sequence of actions, which are tasks such as building code or deploying to test environments. AWS CodePipeline provides you with a graphical user interface to create, configure, and manage your pipeline and its various stages and actions, allowing you to easily visualize and model your release process workflow.

**Parallel Execution**  
You can use CodePipeline to model your build, test, and deployment actions to run in parallel in order to increase your workflow speeds.

**AWS integrations**

AWS CodePipeline can pull source code for your pipeline directly from[AWS CodeCommit](https://aws.amazon.com/codecommit/), [GitHub](https://docs.aws.amazon.com/codepipeline/latest/userguide/pipelines-webhooks-migration.html), [Amazon ECR](https://aws.amazon.com/ecr/), or[Amazon S3](https://aws.amazon.com/s3/). It can run builds and unit tests in[AWS CodeBuild](https://aws.amazon.com/codebuild/). CodePipeline can deploy your changes using[AWS CodeDeploy](https://aws.amazon.com/codedeploy/),[AWS Elastic Beanstalk](https://aws.amazon.com/elasticbeanstalk/), [Amazon Elastic Container Service](https://aws.amazon.com/ecs/) (Amazon ECS), or [AWS Fargate](https://aws.amazon.com/fargate/).

You can model[AWS CloudFormation](https://aws.amazon.com/cloudformation/) actions that let you provision, update, or delete AWS resources as part of your release process. This also enables you to continuously deliver serverless applications built using [AWS Lambda](https://aws.amazon.com/lambda/), [Amazon API Gateway](https://aws.amazon.com/api-gateway/), and[Amazon DynamoDB](https://aws.amazon.com/dynamodb/) with the [AWS Serverless Application Model](https://github.com/awslabs/serverless-application-model) (AWS SAM).

You can also trigger custom functions defined by code at any stage of your pipeline using CodePipeline's [integration with AWS Lambda](http://docs.aws.amazon.com/codepipeline/latest/userguide/how-to-lambda-integration.html). For example, you can trigger a Lambda function that tests whether your web application deployed successfully.

CodePipeline lets you configure a pipeline that ties these services together along with [third-party developer tools](https://aws.amazon.com/codepipeline/product-integrations/) and custom systems.

**Pre-built plugins**

AWS CodePipeline allows you to integrate third-party developer tools, like GitHub or Jenkins, into any stage of your release process with one click. You can use third-party tools for source control, build, test, or deployment. Learn more about our integrations [here](https://aws.amazon.com/codepipeline/product-integrations/).

**Custom plugins**

AWS CodePipeline allows you to integrate your own custom systems. You can register a custom action that allows you to hook your servers into your pipeline by integrating the CodePipeline open source agent with your servers. You can also use the CodePipeline Jenkins plugin to easily register your existing build servers as a custom action.

**Declarative templates**

AWS CodePipeline allows you to define your pipeline structure through a declarative JSON document that specifies your release workflow and its stages and actions. These documents enable you to update existing pipelines as well as provide starting templates for creating new pipelines.

**Access control**

AWS CodePipeline uses AWS IAM to manage who can make changes to your release workflow, as well as who can control it. You can grant users access through IAM users, IAM roles, and SAML-integrated directories.

**Receive Notifications**

You can create notifications for events impacting your pipelines. Notifications will come in the form of [Amazon SNS](https://aws.amazon.com/sns/) notifications. Each notification includes a status message as well as a link to the resources whose event generated that notification.

* **Installation and Working**

If you are new to CodePipeline, Before you can use AWS CodePipeline for the first time, you must complete the following steps

**Step1:** Create an AWS account

**Step2:** Create or use an IAM user

**Step3:** Use an IAM managed policy to assign CodePipeline permissions to the IAM user

**Step4:** Install the AWS CLI

**Step5:** Open the console for CodePipeline

**Step6:** After you complete the steps in [Getting started with CodePipeline](https://docs.aws.amazon.com/codepipeline/latest/userguide/getting-started-codepipeline.html), you can try one of the AWS CodePipeline tutorials

* **Advantages**
* **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 customize the steps to deploy your application and its dependencies.

* **Get started fast**

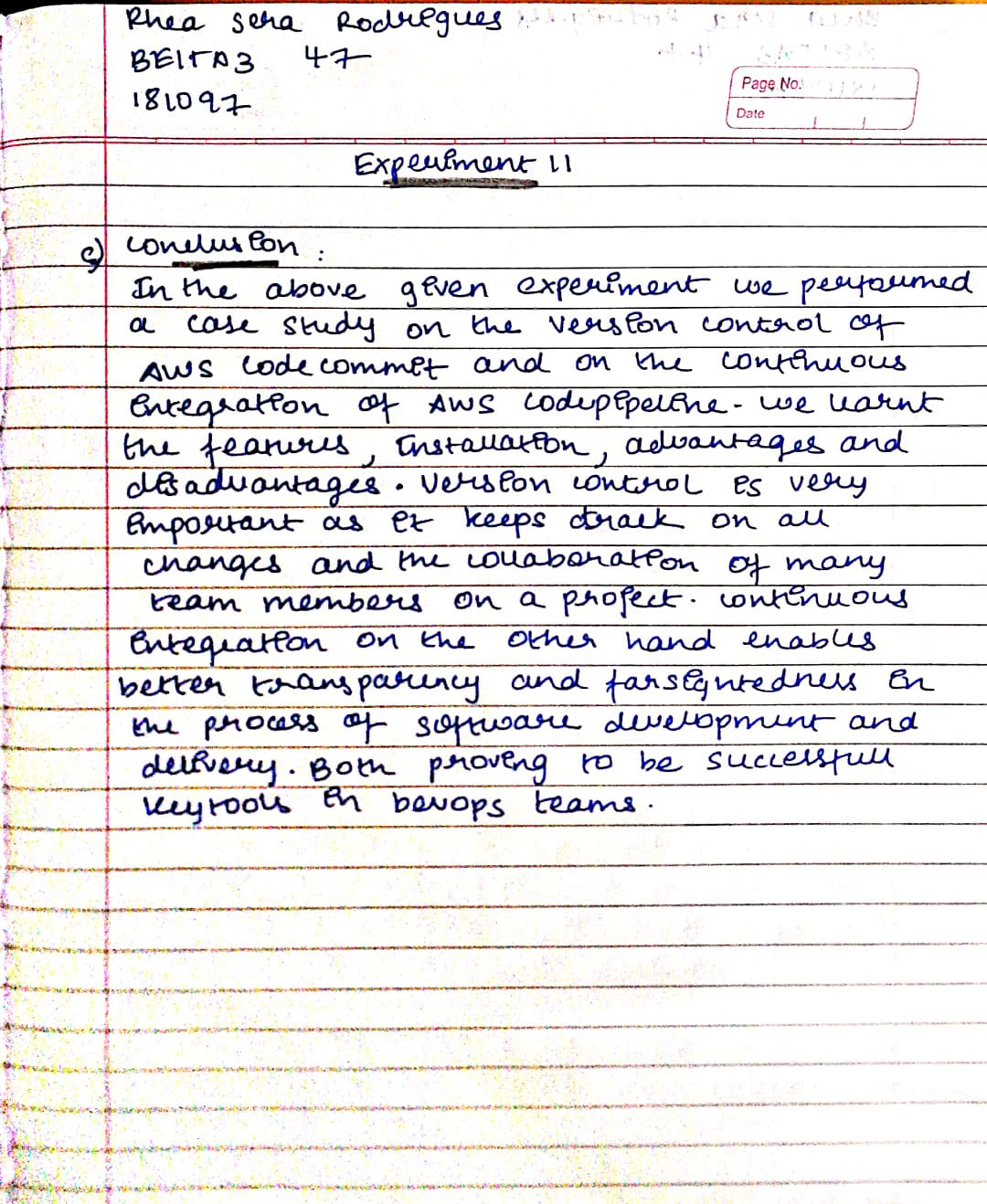
With AWS CodePipeline, you can immediately begin to model your software release process. There are no servers to provision or set up. CodePipeline is a fully managed continuous delivery service that connects to your existing tools and systems.

* **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.

* **Disadvantages**
* The console UI and the overall usability is bad. With CI/CD systems, usability is often the difference between people following practices diligently and not following them.
* It forces unnecessary rules. For example, you have to specify a source, which has to be either S3, AWS CodeCommit, Github or a few others. If you want to pickup the source from a custom location, you have to come up with hacks such as specify a dummy source and then create a codebuild stage to pull the custom source.
* Codepipeline has to be composed of multiple AWS services like CodeCommit, Codebuild, Codedeploy etc. This makes it extremely complicated and overly dependent on AWS .For example, the build stage has to be a CodeBuild module. CodeBuild is another AWS service, which you can assume is similar to a build agent.
* It is a walled garden. The agents will be on AWS, the infra will be on AWS, the CI/CD server is on AWS. It makes all your infra completely dependent on AWS.

**Conclusion:**

****