**Project Allocated by** : Technical Hub

**Mentoring & Monitoring** : M D Shaifu

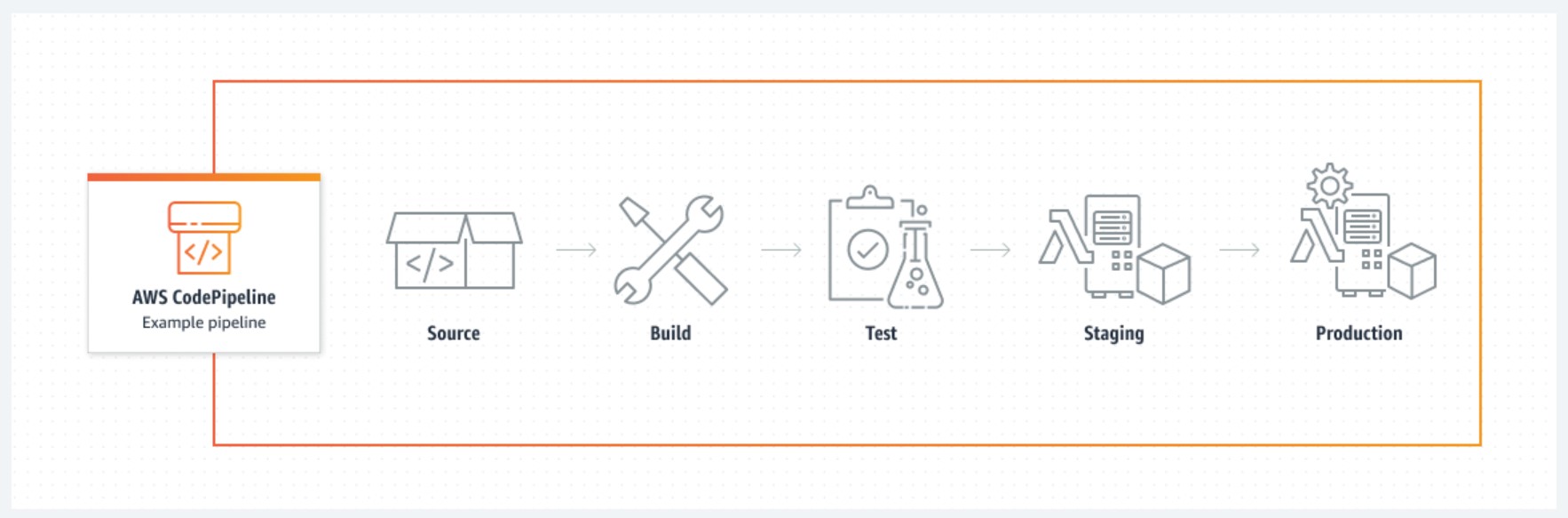
**Task Allocated Time :** 30/06/2024

**Eod :** 026/06/2024

**Team : IT OPERATIONS** ~ Mounika.N(21MH1A05B4), Deepika.K(21MH1A05A1), Tulasi.I(21MH1A05D5), Priya.J(21MH1A0597), M.K.S Amrutha(22MH5A0507)

**Platform Used**: AMAZON WEB SERVICES **SERVICES USED:**

* **AMAZON SIMPLE STORAGE SERVICE ‘**
* **AMAZON CODE PIPE LINE**
* **GITHUB VERSION ONE**
* **ELASTIC BEANSTALK (OPTIONAL)**



GITHUB (HTML - 3 TIER WEB APPLICATION)

||

AWS CODEPIPELINE

||

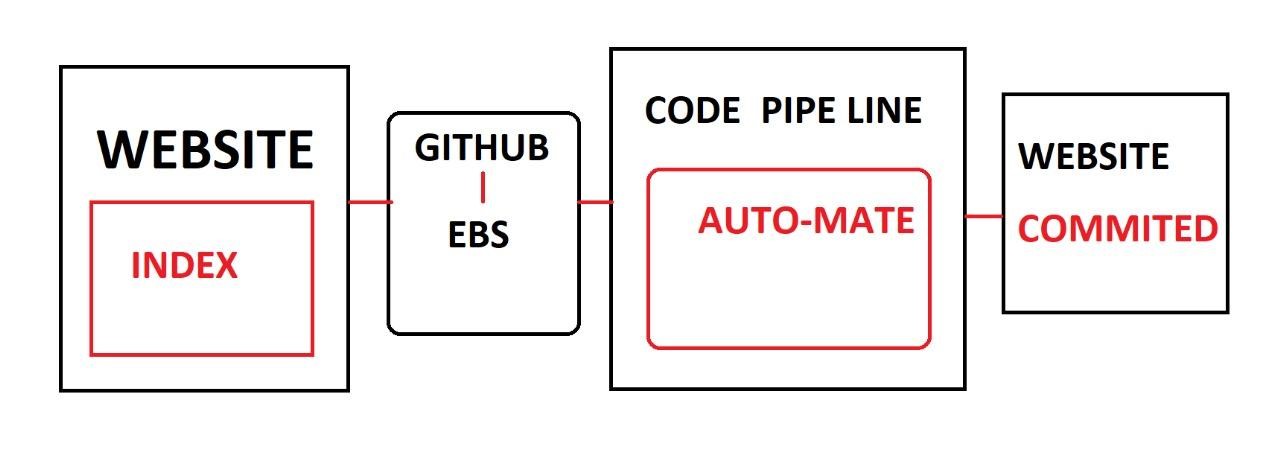
AWS SIMPLE STORAGE SERVICE DEPLOYING STAGE

# Abstract

The abstract outlines the continuous deployment process using AWS CodePipeline. This streamlined approach automates the deployment of applications by integrating source code repositories with build, test, and deployment stages. Leveraging AWS services like CodeBuild and CodeDeploy, the pipeline facilitates consistent and efficient application updates, ensuring rapid and reliable delivery. This abstract highlights the significance of implementing continuous deployment using AWS CodePipeline to enhance software development and deployment workflows.

Team : Cloud Surferes

**Task Plan :**



Source used : GitHub (Posted an 3 -Tier web application on GitHub and naked it Public )

Pipe : AWS code PIPELINE to detect the changes in source and to update it in deployment

Deployment : Amazon simple Storage service

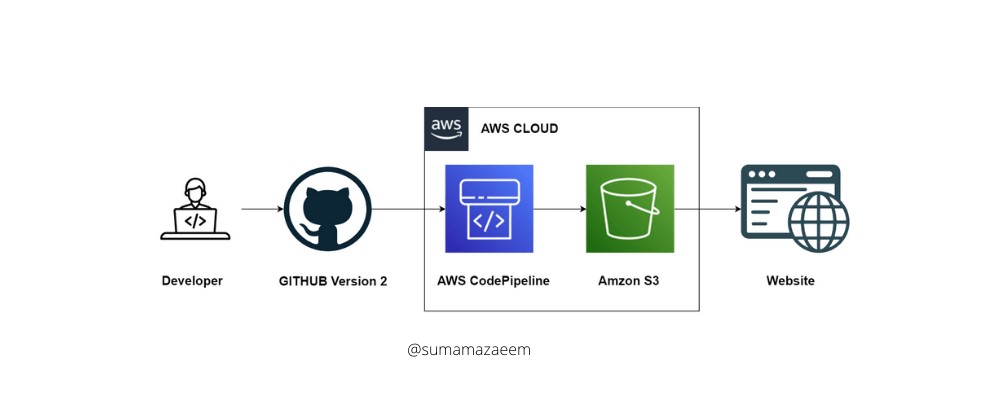
By orchestrating source code changes through various stages, including build, test, and deployment, CodePipeline enables efficient and reliable software updates. Leveraging AWS services such as CodeBuild and CodeDeploy, the process ensures consistency and minimizes manual intervention. This abstract underscores the significance of adopting CodePipeline to streamline deployment workflows and accelerate the delivery of applications.



**GITHUB :**

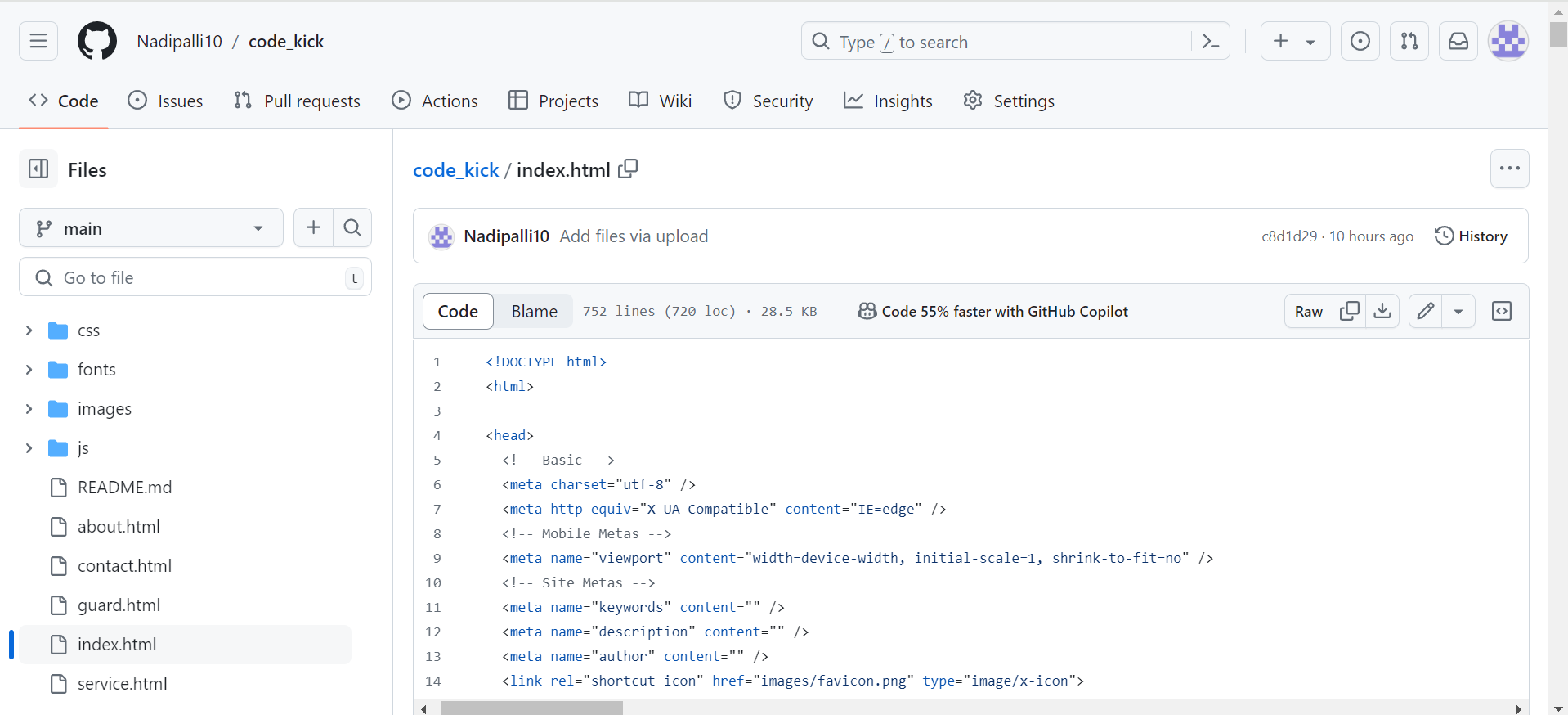
GitHub is a widely used web-based platform for version control and collaboration on software development projects. It allows developers to store and manage their source code, track changes over time, and collaborate with others through features like pull requests and code reviews. GitHub provides a user-friendly interface for creating and managing repositories, which are used to organize and store project files. Developers can clone repositories locally, make changes, and then push those changes back to the remote repository. GitHub also supports issue tracking, wikis, and project boards to facilitate project management. It plays a crucial role in open-source software development by providing a centralized platform for sharing code and collaborating with a global community.

**Attaching Github as an source provider in AWS :**



GitHub can be seamlessly integrated into AWS CodePipeline as a source provider, facilitating a streamlined and automated software delivery pipeline. This integration leverages the strengths of both platforms to enhance the development and deployment process. Here's how GitHub is attached to AWS CodePipeline:

* **Source Stage Selection**: In the AWS CodePipeline setup, developers can choose GitHub as the source provider when configuring their pipeline.
* **GitHub OAuth**: Developers authenticate their GitHub accounts with AWS using OAuth tokens to establish a secure connection between GitHub and CodePipeline.
* **Repository Selection**: Users can select the specific GitHub repository they want to integrate with the pipeline.
* **Branch and Trigger**: Developers can specify the branch to be monitored for changes. Whenever code changes are pushed to this branch, GitHub sends webhook notifications to CodePipeline.
* **Webhook Setup**: CodePipeline sets up a webhook in the GitHub repository, enabling real-time notifications of code changes.
* **Automatic Pipeline Execution**: When changes are detected in the specified GitHub branch, the CodePipeline automatically triggers the pipeline execution.
* **Source Code Retrieval**: CodePipeline fetches the latest source code from the GitHub repository and prepares it for further stages.
* **Customization**: Users can configure the pipeline to perform specific actions on the source code, such as building, testing, and deploying.
* **Parallel and Sequential Stages**: CodePipeline allows developers to design stages that execute in parallel or sequentially, tailoring the deployment process to their needs.
* **Feedback and Monitoring**: Developers receive feedback on each stage's execution, including success, failure, or warnings, via the pipeline's dashboard.
* **Rollback**: If any stage fails, CodePipeline can automatically initiate a rollback to a previously successful version of the application.
* **Version Tracking**: CodePipeline tracks the versions of the source code and artifacts through the pipeline stages, providing a clear history of changes.
* **Multiple Environments**: Developers can set up multiple pipeline stages to deploy to different environments, such as development, testing, and production.
* **Approval Gates**: CodePipeline can be configured to include manual approval steps before deploying to critical environments, ensuring controlled releases.
* **Artifact Storage**: CodePipeline stores the artifacts produced at each stage, enabling traceability and auditing.
* **Scaling**: Both GitHub and AWS CodePipeline are scalable platforms, ensuring that they can handle projects of varying sizes and complexities.
* **Collaboration**: Developers can collaborate on code changes using GitHub's pull requests and code review features before code reaches deployment stages.
* **Third-Party Integrations**: CodePipeline supports integration with other AWS services and third-party tools, expanding its capabilities beyond the source-to-deploy cycle.
* **Infrastructure as Code**: Developers can define their pipelines using Infrastructure as Code (IaC) tools like AWS CloudFormation, enhancing repeatability and automation.
* **End-to-End Visibility**: The integration offers end-to-end visibility into the software delivery process, from source code changes to deployment results.
* **CI/CD Best Practices**: Combining GitHub's version control and collaboration features with CodePipeline's automation fosters the implementation of CI/CD best practices.
* **Security**: AWS provides security features to protect code and artifacts throughout the pipeline, while GitHub offers authentication and access control mechanisms.
* **Continuous Improvement**: As both platforms evolve, the integration continues to improve, offering new features and capabilities to enhance the development process.
* **Documentation and Support**: Both GitHub and AWS offer extensive documentation and community support to help developers effectively integrate and utilize the platforms.
* **Cost-Effectiveness:** GitHub's free and paid plans, along with AWS's pay-as-you-go odel, provide cost-effective options for businesses of all sizes.
* **Hybrid Environments**: GitHub integration with CodePipeline supports hybrid environments where components can reside on-premises and in the cloud.
* **Quick Setup**: Integrating GitHub as a source provider in CodePipeline requires minimal configuration, enabling developers to get started swiftly.
* **Feedback Loop**: Integrating GitHub and CodePipeline enhances the feedback loop by automating code delivery, enabling developers to iterate faster.
* **Real-Time Updates**: GitHub's webhooks and CodePipeline's event-driven architecture ensure real-time updates are captured for continuous delivery.
* **Ecosystem Synergy**: Integrating two powerful platforms like GitHub and AWS CodePipeline creates a synergy that fosters efficient, automated, and reliable software delivery processes, benefitting developers and end-users alike.



**AWS CODEPIPELINE :**

AWS CodePipeline is a fully managed continuous integration and continuous delivery (CI/CD) service that automates the process of building, testing, and deploying code changes. It allows developers to define multi-stage pipelines that automate the release process for applications, enabling rapid and reliable software delivery. CodePipeline integrates with various AWS services like CodeBuild and CodeDeploy to provide a seamless end-to-end CI/CD workflow. With support for source providers like GitHub and AWS CodeCommit, it enables automatic triggers on code changes, ensuring efficient and consistent deployments. CodePipeline's visual interface allows easy pipeline creation and management, while its event-driven architecture ensures real-time updates throughout the pipeline stages.

**STEPS TO CREATE AN CODEPIPELINE AND TO ATTACH SOURCE** :

**Sign In to AWS Console**: Log in to your AWS Management Console using your credentials.

**Open AWS CodePipeline**: Navigate to the AWS CodePipeline service from the console's dashboard.

**Create Pipeline**: Click on the "Create pipeline" button to start the pipeline creation process.

**Pipeline Settings**: Enter a name for your pipeline Optionally, provide a description.

**Source Stage**:Choose your source provider (e.g., GitHub) and connect to your account.select the repository and branch you want to use as the source.Configure any other relevant settings, like whether to detect changes based on webhooks or a schedule.

**Build Stage (Optional)**:If needed, add a build stage using AWS CodeBuild or another compatible build service.Configure build settings, including build environment, commands, and artifacts.

Deployment Stage:Add a deployment stage using AWS CodeDeploy or another compatible deployment service.Configure deployment settings, such as target environment and deployment actions.

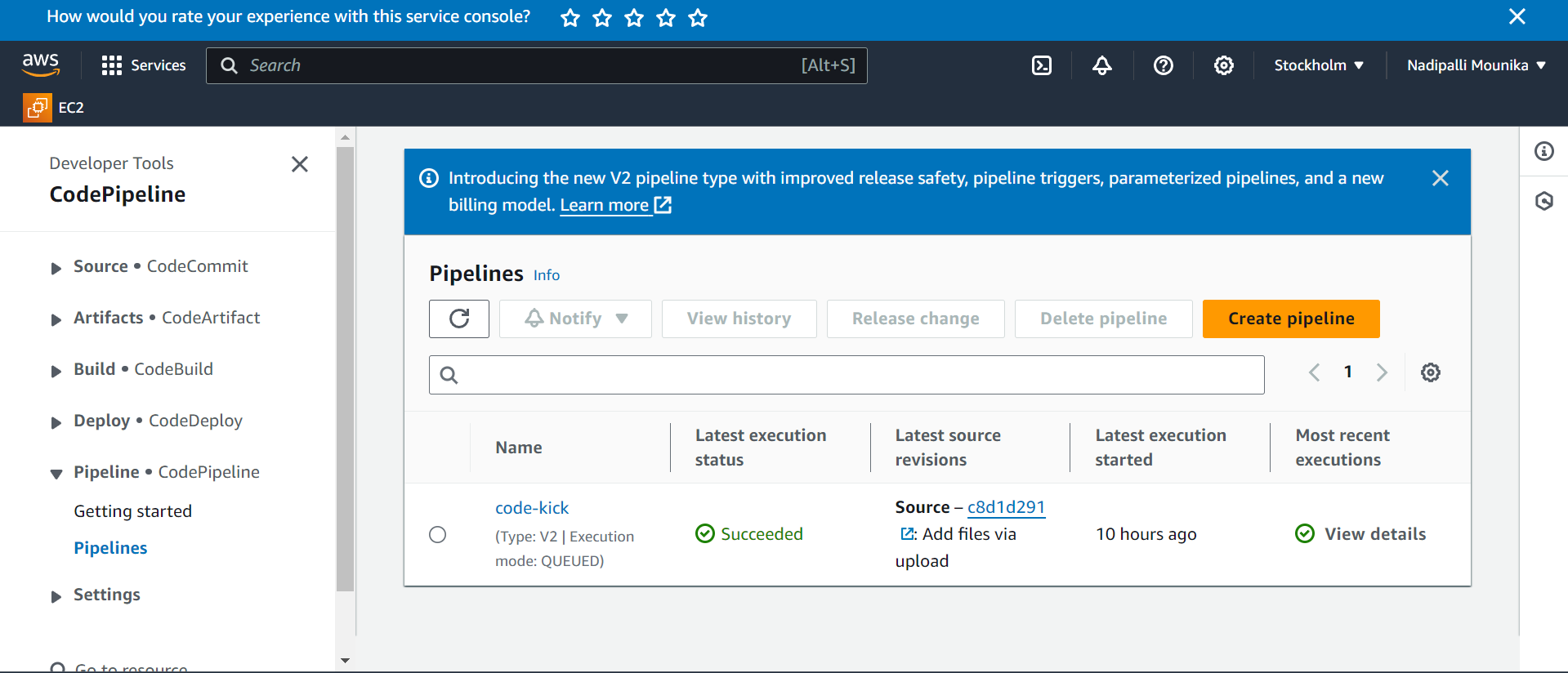
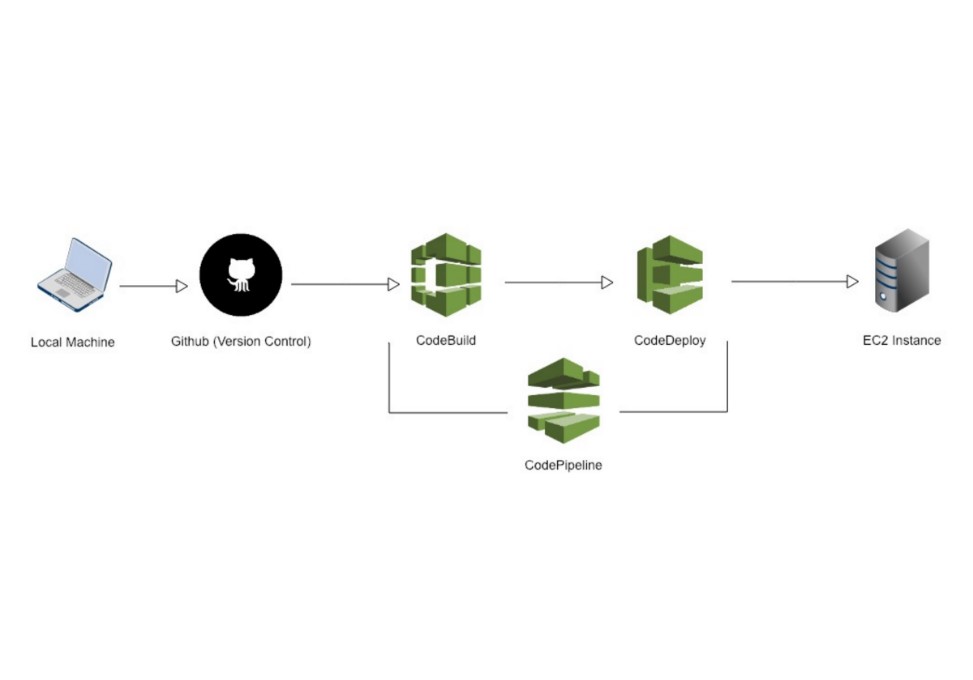
**Add More Stages (Optional)**:You can add additional stages for testing, approval, or any other necessary steps.

**Configure Artifacts**:Specify how artifacts are passed between stages. These can be source code, build outputs, or any other relevant files.

**Review and Create**:Review the pipeline configuration to ensure everything is set correctly. Click on "Create pipeline" to create the pipeline.

**Pipeline Execution**:The pipeline will automatically execute based on the configuration and trigger conditions you've set.

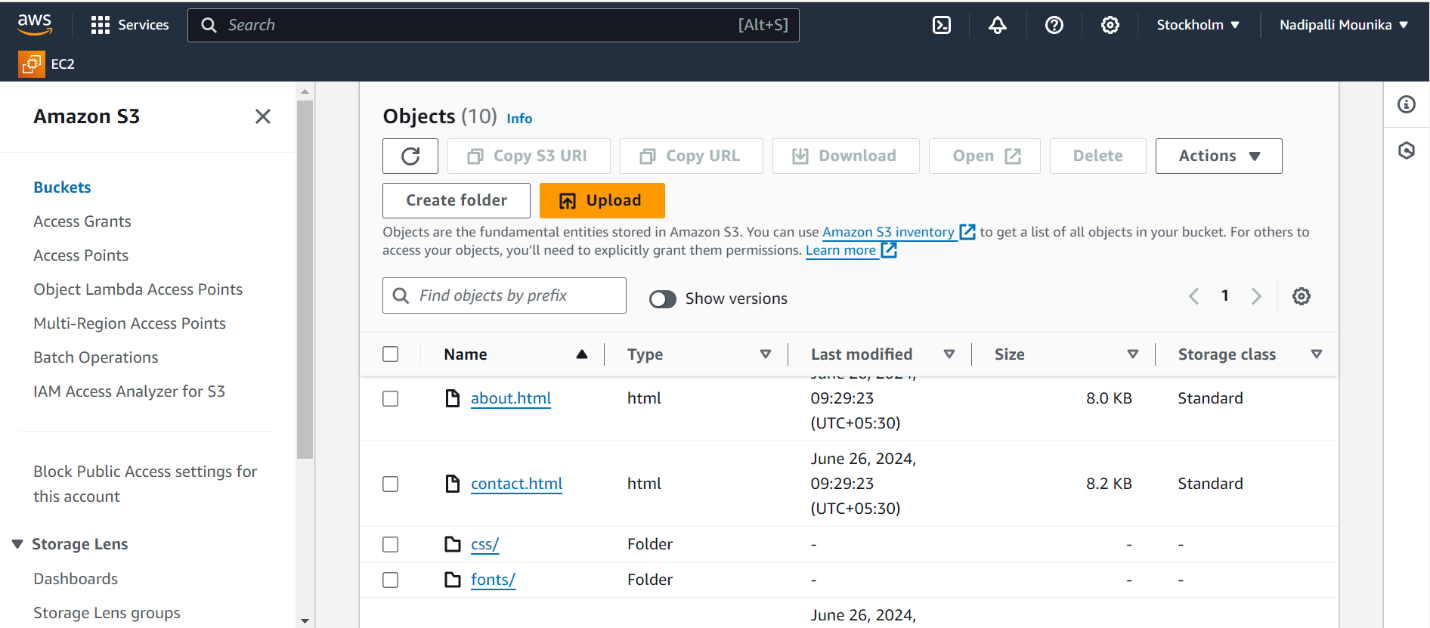
**Monitor Progress**:Monitor the progress of your pipeline in the AWS CodePipeline console. View details about each stage's execution and any potential issues.By following these steps, you'll create an AWS CodePipeline that integrates with your chosen source provider, such as GitHub, to automatically trigger pipeline executions whenever there are code changes. This streamlined process enables efficient and automated software delivery from your Git repository to the desired deployment environment. Remember that the exact steps and options may vary based on the current interface and features provided by AWS at the time of your usage.



**Deployment in S3** :

Amazon S3 (Simple Storage Service) is a highly scalable and durable object storage service provided by Amazon Web Services (AWS). It allows businesses to store and retrieve vast amounts of data, such as images, videos, backups, and application assets, in a simple and cost-effective manner. S3's architecture is designed for high availability, providing 99.999999999% (11 nines) durability for stored objects across multiple Availability Zones. With its pay-as-you-go pricing model, users only pay for the storage they use, making it a versatile and cost-efficient solution for organizations of all sizes.

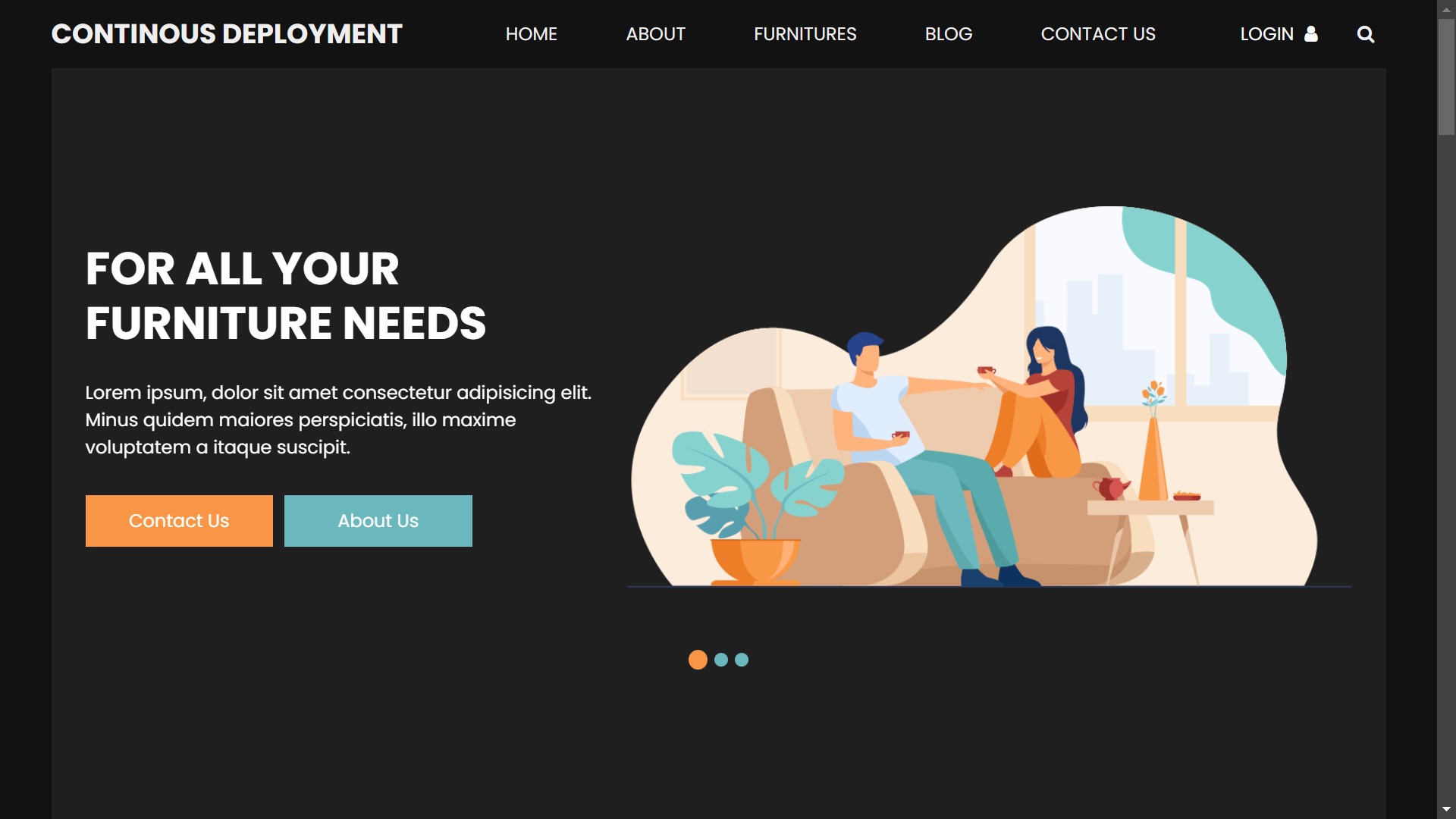
Deploying an S3 bucket using AWS CodePipeline involves creating a pipeline that automates the deployment process



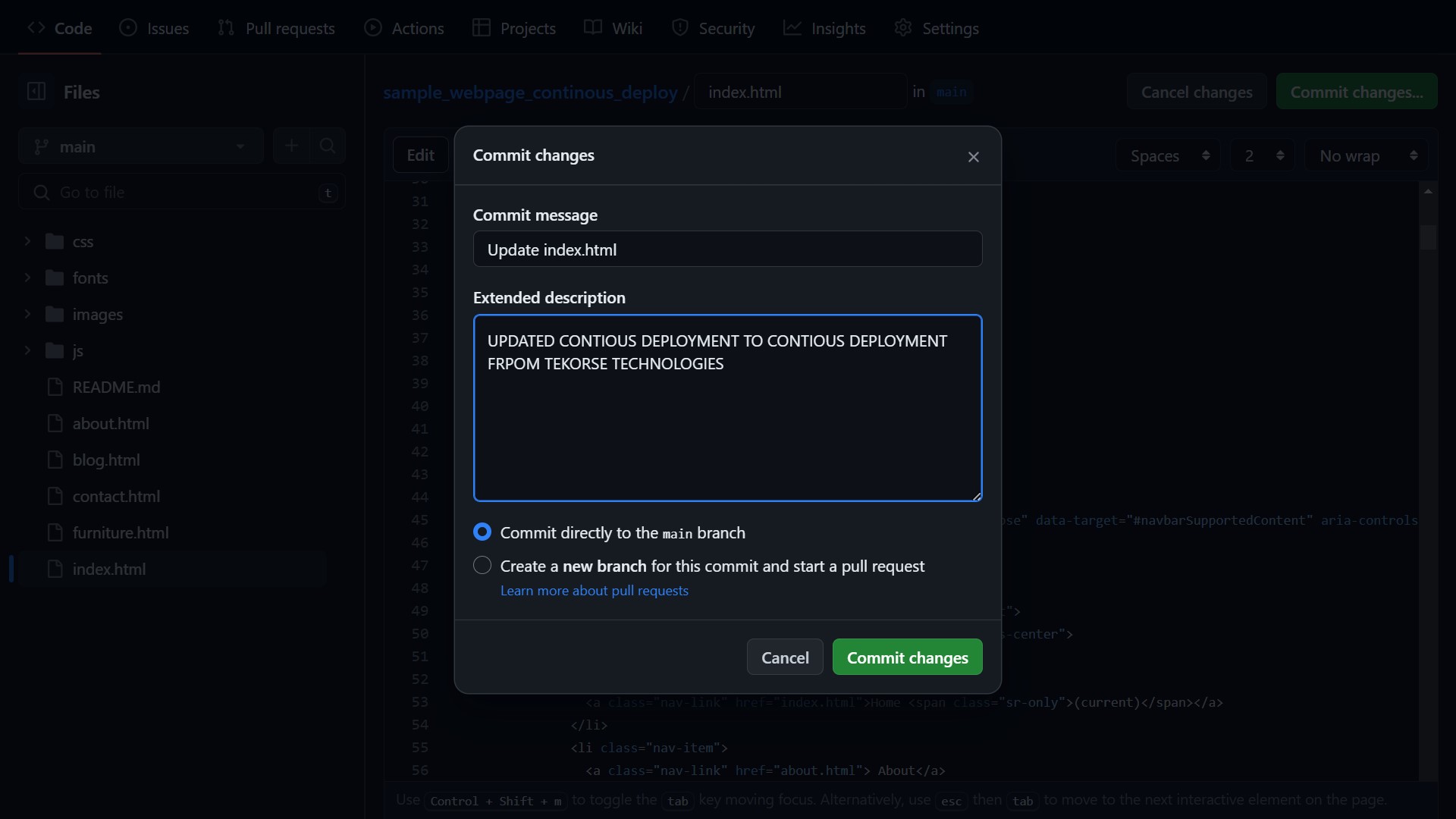
STAGES OF TASK

1. INITIAL SAMPLE 3 TIER WEB APPLICATION
2. COMMIT CHANGES IN GIT
3. NOTICE CHANGES IN PIPELINE
4. DEPLOYMENT IN S3

1.



**2.**



**3.**

