DevOps Introduction: Working with Cl CD and Continuous Monitoring Tools

LEARNER'S GUIDE

Our Essence

Transforming People

We empower people to gain work ad life skills, transforming their lives, improving their employability and inspiring them to grow through lifelong learning

Vision

To be the leader and trusted lifelong partner in Continuing Education and Training

Mission

To provide learning that makes Every Worker a Better Worker, Every Job a Better Job, Every Company a Better Company

Our Values

People are our priority

- Respect and teamwork
- Listen to customer needs

Passion for our goals

- Passion for lifelong learning
- Transforming people through learning
- Heart and hunger to serve our customers

Performance is our business

- Deliver exceptional / sustainable value
- Unmatched choice for our people and customers

Table of Contents

LESSON 4: TOOLS AND PRACTICES FOR CONTINUOUS MONITORING

LESSON 5: AUTOMATING DEPLOYMENT WITH INFRASTRUCTURE AS CODE (IAC)

LESSON 6: INTEGRATING SECURITY AND COMPLIANCE WITH CI/CD

LESSON 7: COLLABORATIVE CI/CD PIPELINE MANAGEMENT

LESSON 8: CI/CD AND CONTINUOUS MONITORING SUCCESS STORIES

LESSON 9: FUTURE OF DEVOPS: EMERGING TRENDS AND INNOVATIONS

INTRODUCTION TO THE COURSE

Overview

This Learner's Guide aimed to present the content and activities aligned to the Workforce Skills Qualification (WSQ) Offer Customised and Personalised Service module.

Target Audience

This course has been specifically designed for the learners with a background in the retail or transactional (over-the-counter) services industries. The course content and examples are thus slanted towards these sectors. If you are not from these sectors, please advise your trainers.

Assumed Skills and Knowledge

Assumed skills and knowledge means what we expect you to know and be able to do already, before you start the course. We assume you have:

- Basic knowledge of CI/CD
- Basic Knowledge of Github
- Basic knowledge of DevOps software setup

Course Objectives

At the end of this course, learners will be able to:

- Have the knowledge and application skills to proactively offer and promote service to customers.
- Be able to access and communicate detailed dish/product and service information sought by a diverse range of customers in order to make recommendations that meet customers' personal needs.

Course Components

Workshops

Final Assessment	
Assessment Method	**Duration**
Written Assessment	30 minutes
Practical Assessment	150 minutes

Course Duration

Classroom Training	Final Assessment	Total
21 hours	3 hours	24 hours

Warm-Up and Introduction

"This Is Service" Icebreaker Activity

Choose 1 item which best represents what service is in the retail industry from a service provider perspective?

Share the following with your group and then the class:

1. Your name
2. Company you are from
3. What do you do at your workplace
4. Why did you choose that particular item to represent service in the retail sector

Notes

Module 1: Unveiling DevOps Culture and Practices

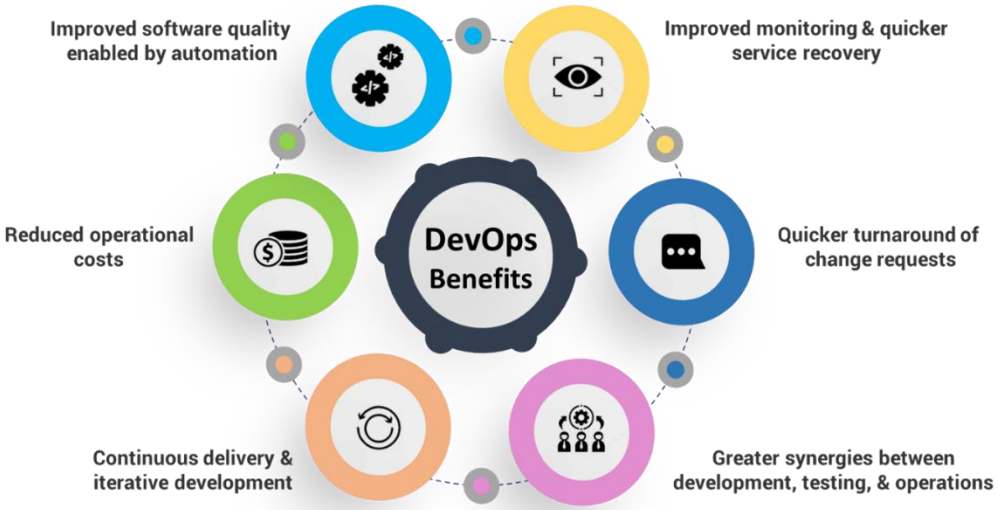
Defining DevOps and its role in modern software development

What is DevOps?

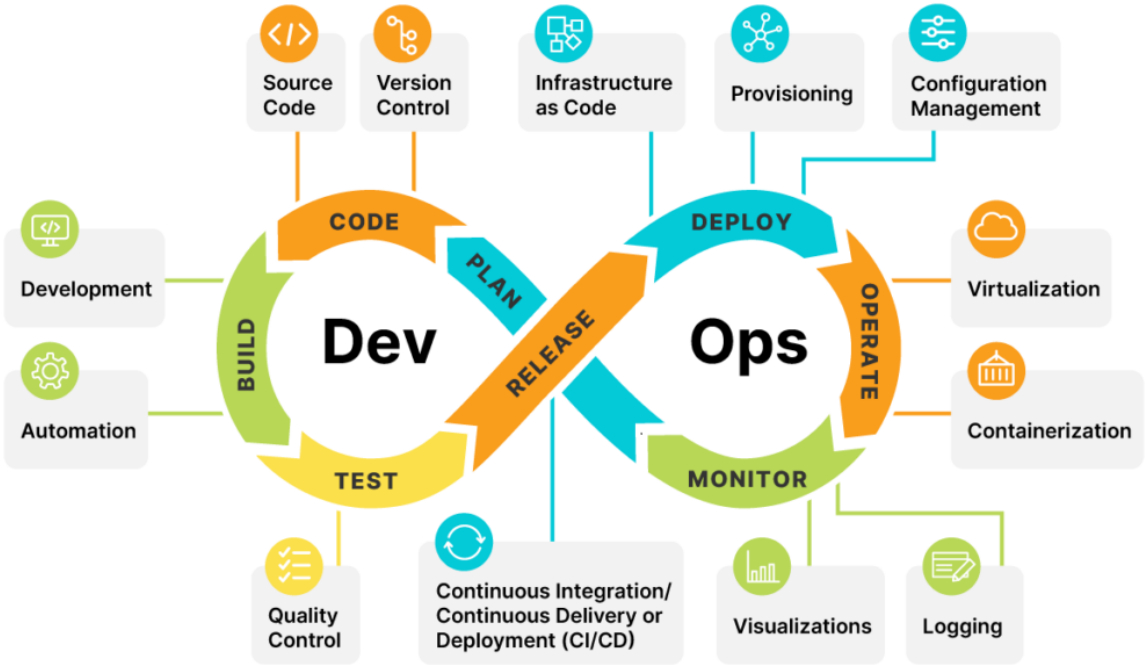
DevOps, short for Development and Operations, is a set of practices and cultural philosophies. That aims to bridge the gap between software development and IT Operation. DevOps brings together development, testing, deployment, and operations under one umbrella. That enables organizations to achieve continuous integration and continuous delivery (CI/CD) and maintain a rapid development pace.

- Enable rapid evolution of products or services
- Reduce risk, improve quality across the portfolio, and reduce costs
- DevOps integration targets product delivery, quality testing, feature
- development, and maintenance releases to improve reliability and
- security and faster development and deployment cycles.
- The adoption of DevOps is being driven by factors such as:
- Use of agile and other development processes and methodologies
- Demand for an increased rate of production releases from application and business stakeholders
- Wide availability of virtualized and cloud infrastructure from internal and external providers
- Increased usage of data center automation and configuration management tools

Why DevOps? – Delivery challenges



DevOps Lifecycle



Plan:

- During this phase, teams define project goals, gather requirements, and create a roadmap for development. Key activities in the planning phase contain requirement

gathering, scope definition, project estimation, and establishing project milestones and timelines.

Code:

- The developers will write the code and prepare it for the next phase during the coding stage. Developers will write code in accordance with the specifications outlined in the planning phase and will ensure that the code is created with the project's operations in mind.

Build:

- Code will be introduced to the project during the construction phase, and if necessary, the project will be rebuilt to accommodate the new code. This can be accomplished in a variety of ways, although GitHub or a comparable version control site is frequently used.
- The developer will request the addition of the code, which will then be reviewed as necessary. The request will be approved if the code is ready to be uploaded, and the code will be added to the project. Even when adding new features and addressing bugs, this method is effective.

Release:

- The release phase occurs when the code has been verified as ready for deployment and a last check for production readiness has been performed. The project will subsequently enter the deployment phase if it satisfies all requirements and has been thoroughly inspected for bugs and other problems.

Test:

- The testing phase is essential to ensure the quality and functionality of the software being developed. Rigorous testing is performed to identify and rectify any bugs, errors, or issues. This phase includes various testing types such as unit testing, integration testing, and system testing.

Deploy:

- The deployment phase involves releasing the software to the production environment for end-users to access and utilize. During this phase, deployment processes are automated to ensure consistency and reliability. Key activities in the deployment phase include environment setup, release coordination, deployment automation, and monitoring the deployment process.

Operate:

- The operation phase focuses on the ongoing management and maintenance of the software in the production environment. This phase involves monitoring the software's

performance, addressing user feedback, and handling any issues or incidents that arise. Continuous monitoring and feedback loops are established to ensure optimal software performance and user satisfaction.

Monitor:

- During the monitoring phase, product usage, as well as any feedback, issues, or possibilities for improvement, are recognized and documented.
- This information is then conveyed to the subsequent iteration to aid in the development process. This phase is essential for planning the next iteration and streamlining the pipeline's development process

Key Concepts of DevOps

- Agile Development
- Continuous Integration (CI)
- Continuous Delivery (CD)
- Infrastructure as Code (IaC)

Agile Development

Agile Development has become a cornerstone of successful software delivery. That enables organizations to respond quickly to changing market needs.

Agile Development focuses on iterative and incremental development. By embracing agile principles, organizations experience enhanced flexibility, improved transparency, and increased customer satisfaction.

Continuous Integration (CI)

Continuous Integration (CI) is a vital DevOps practice that aims to merge code changes from multiple developers into shared storage regularly. CI promotes early and frequent code integration. Even automated testing and immediate feedback

increase their popularity. By implementing CI, organizations reduce integration issues. With CI detecting defects earlier in the development process becomes easy. That improves collaboration among development teams.

Continuous Delivery (CD)

Continuous Delivery (CD) goes hand in hand with CI, enabling organizations to deliver software in a reliable and automated manner. The CD focuses on automating the entire release process, from building and testing to deployment and

production. With CD, organizations gain the ability to release software updates swiftly. With minimal manual intervention, and maintain a continuous flow of valuable features to end-users.

Infrastructure as Code (IaC)

A revolutionary concept in DevOps is Infrastructure as Code (IaC). IaC allows organizations to automate infrastructure setup and management, enabling reproducibility, scalability, and consistency across environments. While leveraging IaC tools and practices, organizations gain better control over their infrastructure, reduce human error, and achieve greater efficiency in software deployment.

Understanding the pillars of DevOps

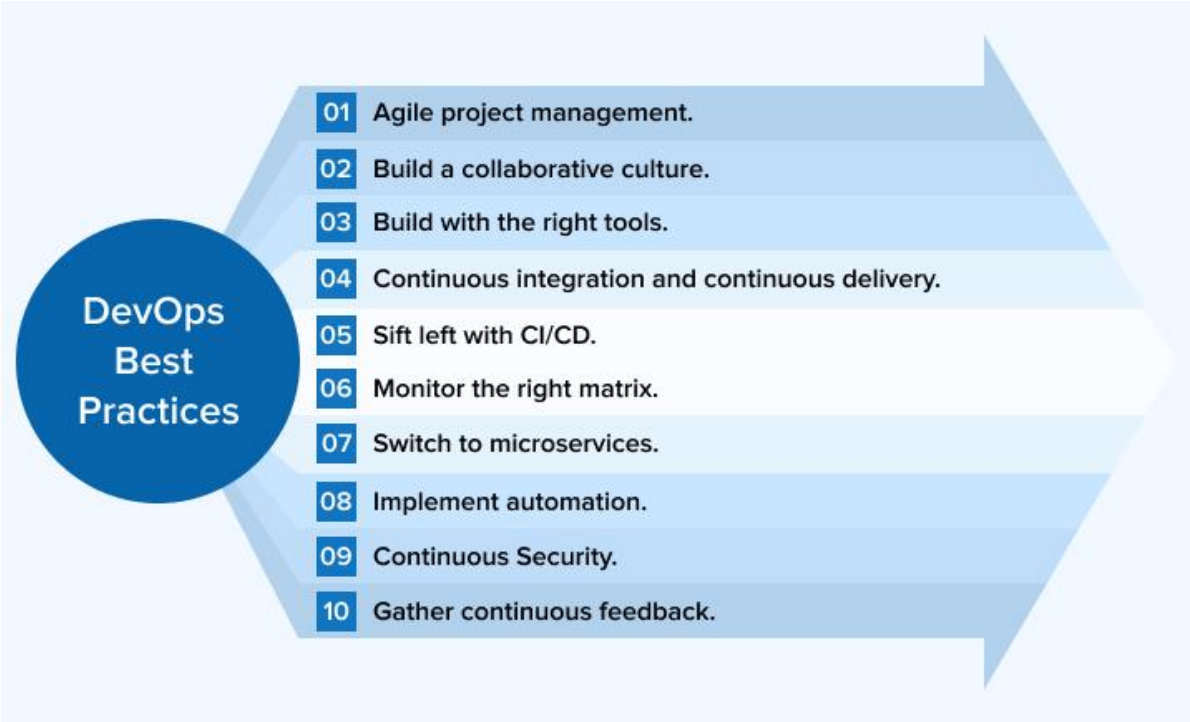
1. **Culture**: Culture is the most important principle of DevOps. It refers to the set of principles, principles and practices that guide teamwork. Teams make decisions and focus on delivering value to customers in a DevOps culture. This culture fosters a sense of shared ownership and supports experimentation and innovation.

2. **Automation:** The second pillar of DevOps is automation. This is the term used to describe the automation of various software development, testing, deployment, and infrastructure management processes using tools and procedures. Teams can minimize errors, improve consistency, and increase productivity by automating these processes.

3. **Measurement:** The third pillar of DevOps is measurement. This means using statistics and data to monitor and improve performance. Teams can identify areas for improvement and make data-driven decisions by monitoring key performance indicators (KPIs). This makes it easier to ensure that everyone is working towards the same goals and things move forward.

4. **Sharing:** Sharing is the fourth pillar of DevOps. This means the knowledge, best practices, and resource sharing that takes place between teams. This way, teams can learn from each other's experiences and always work with the latest information.

Benefits of Implementing DevOps Practices for Efficient Delivery



Module 2:Continuous Integration and Continuous Deployment (CI/CD) Concepts

CI/CD is a method that allows DevOps teams to deliver code updates frequently, reliably, and quickly using continuous integration (CI) and continuous delivery CD) practices. CI/CD emphasizes automation throughout the development lifecycle (Buid, Test, Deploy). By replacing the manual efforts of traditional development, code releases can happen more frequently, and with less bugs and security vulnerabilities.

Exploring the CI/CD Pipeline and its Components

What Is CI (Continuous Integration)?

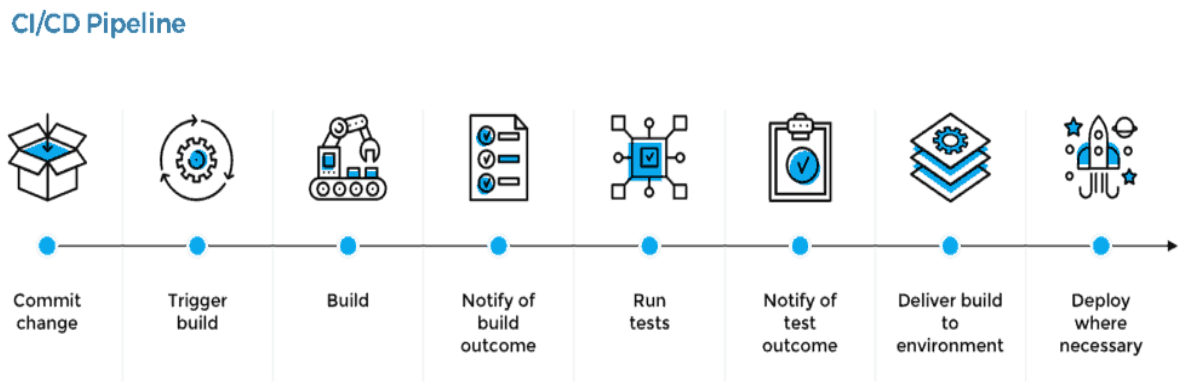
Continuous integration integrates code changes into a shared repository of source code, and it usually automatically tests these changes as soon as they are committed. Continuous integration is performed as your developers write code, rather than afterward.

What Is CD (Continuous Delivery)?

Continuous delivery is an extension of continuous integration in which your team automatically deploys new code to a repository (such as GitHub). It can then be deployed to production at any time based on your needs and the needs of your clients. The added automation used by continuous delivery greatly reduces the amount of effort required to deploy code. With continuous delivery, you can respond more quickly to anything from changes in the market to security flaws.

The CI/CD Pipeline

- The CI/CD pipeline is the set of processes that drive your preferred combination of continuous integration, delivery and deployment, and it forms the base of your DevOps team's operations. In general, the head of your DevOps team is the one in charge of ensuring that the pipeline is working properly.



Components of a CI/CD Pipeline

Every CI/CD pipeline will look different based on your team's specific needs and resources, but in general, they have four essential stages.

Build:

- In this stage, you pull your source code from a repository and build its components into a binary artifact. Your chosen integrated development environment (IDE) may help your developers automate this process.

Test:

- With the CI/CD pipeline, you want to employ as much continuous testing as possible. Unit Testing helps to verify that new features are working as intended, and most of your testing should be this type. Regression testing makes sure that new additions to your code won't break your existing infrastructure.

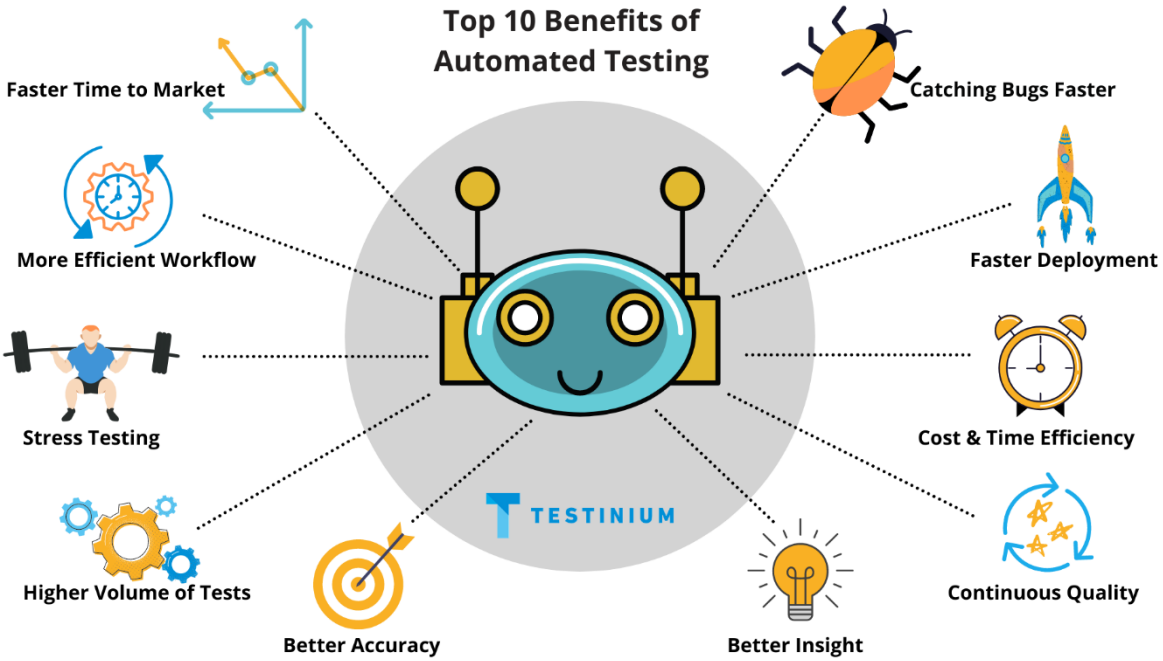
Delivery:

- After your tests are done, your developers' code should go to a staging environment. This enables you to perform A/B tests and find lingering problems, as well as letting your QA team know what they need to look at.

Deploy:

- Once your build has passed all automated testing, it can be deployed in production. With continuous delivery, the build is sent to a human for manual approval, while with continuous deployment the build is deployed automatically.

Understanding the Importance of Automated Testing and Deployment



Implementing CI/CD with tools like Jenkins, GitLab CI/CD

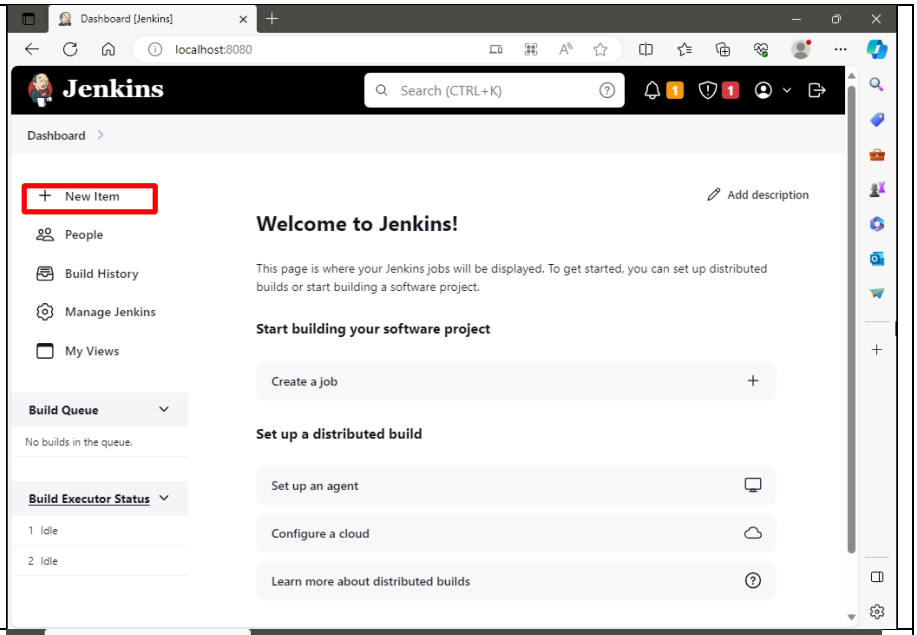
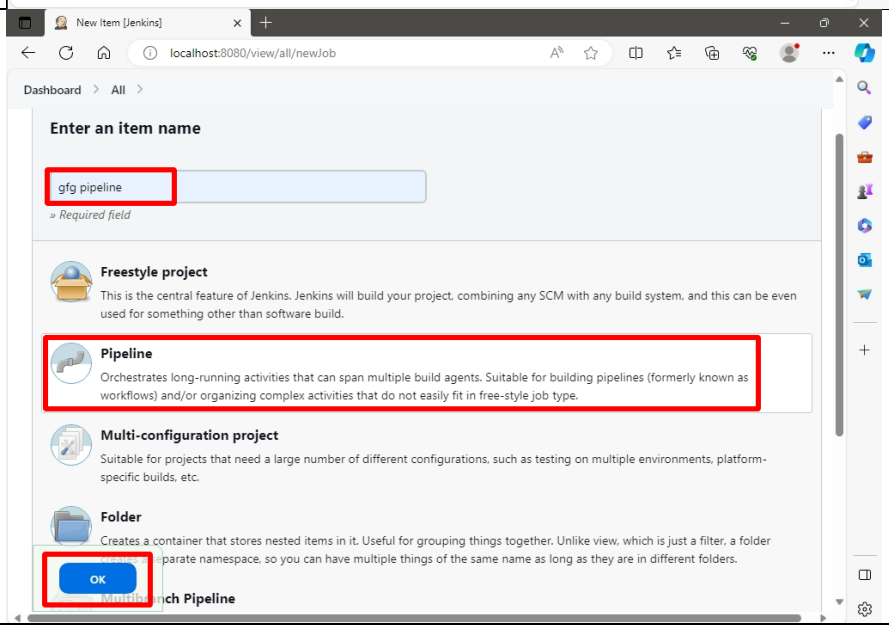
Jenkins

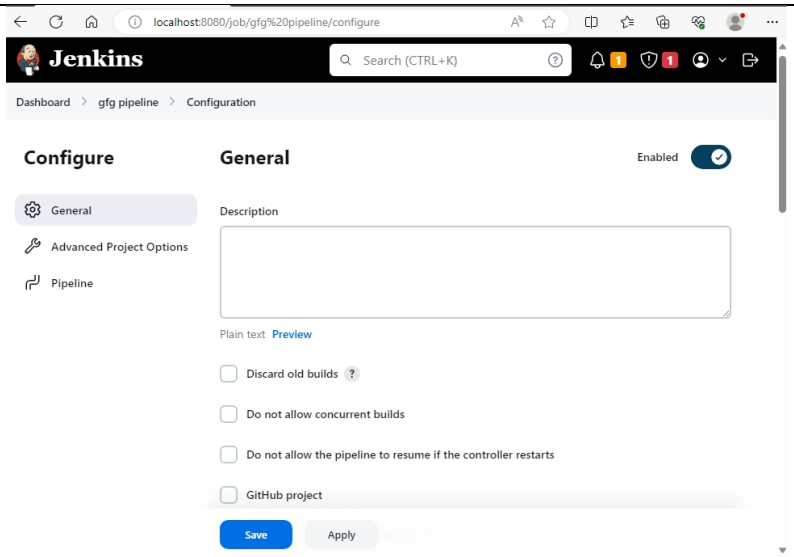
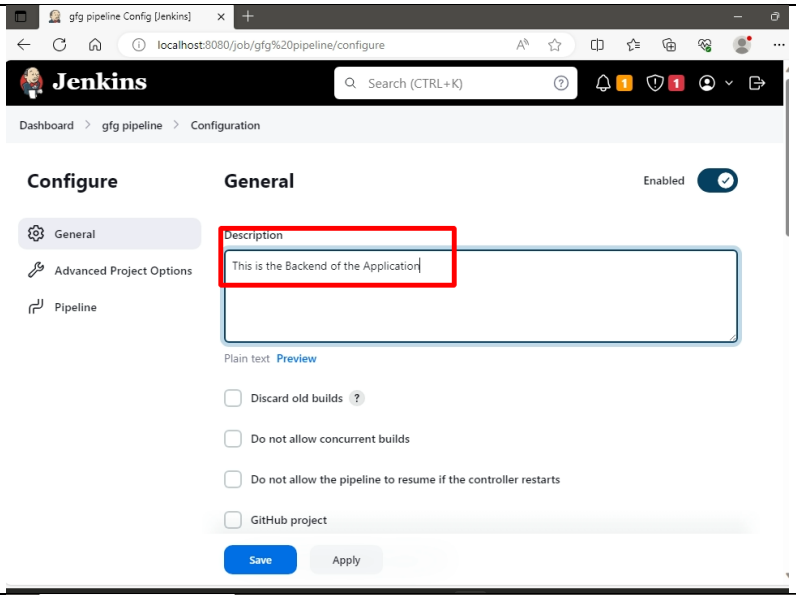
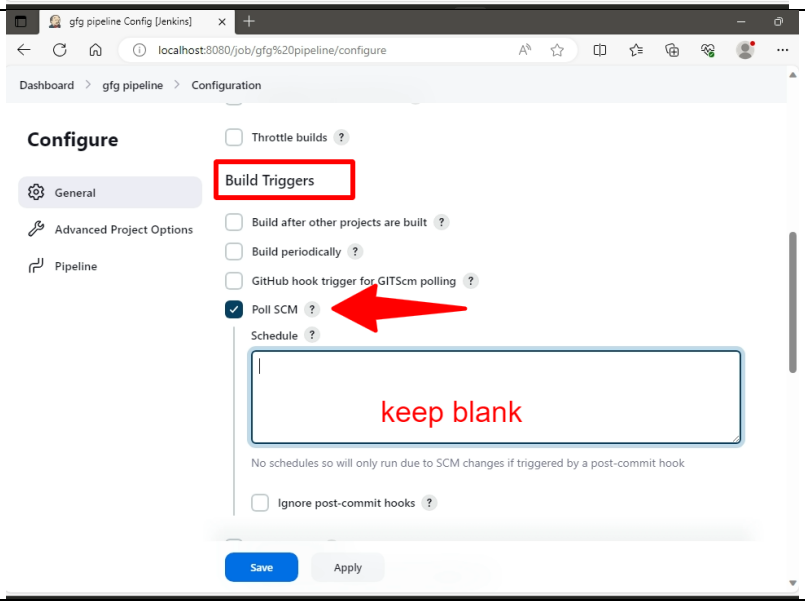
- Jenkins is an open-source automation server in which the central build and continuous integration process take place. It is a self-contained Java-based program with packages for Windows, macOS, and other Unix-like operating systems. With hundreds of plugins available, Jenkins supports building, deploying, and automating software development projects.

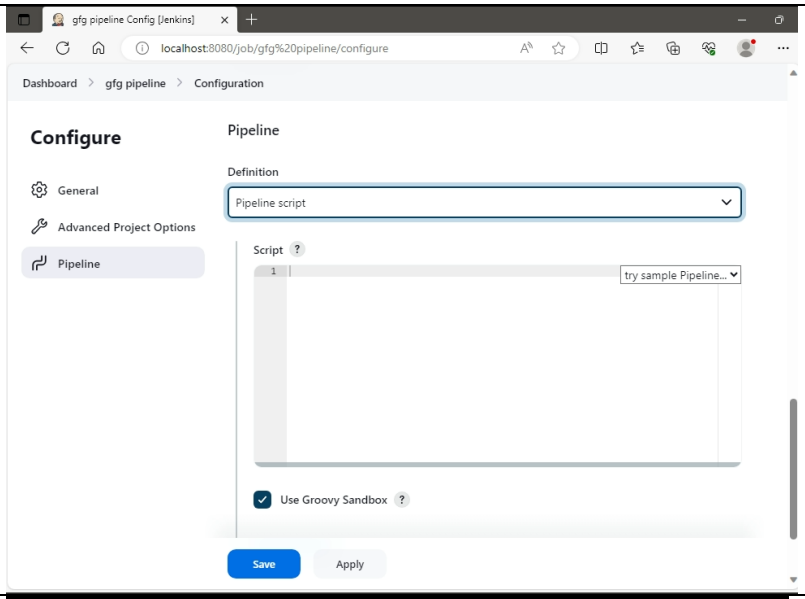
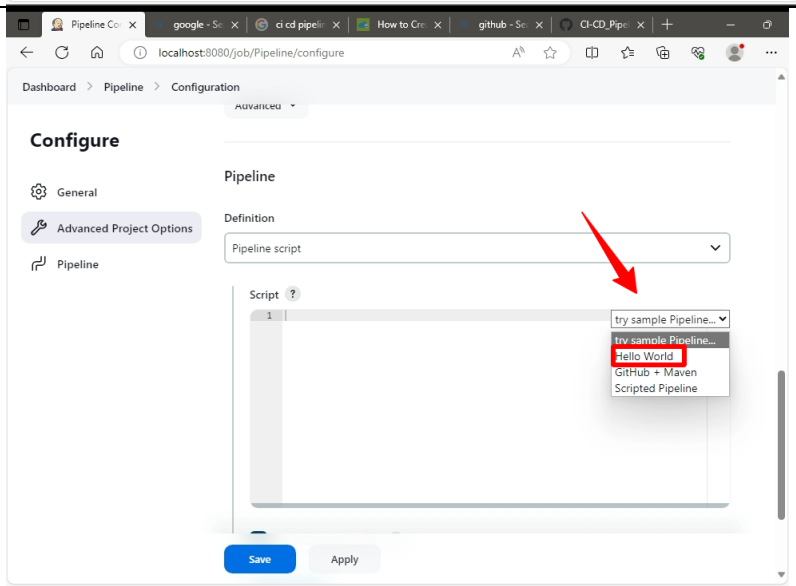
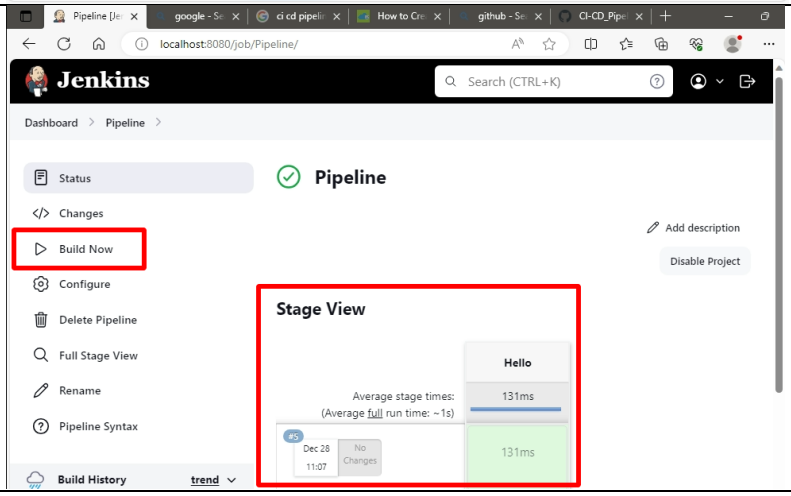
Jenkins key features:

- Easy installation and upgrade on various OSs
- Simple and user-friendly interface
- Extensible with huge community-contributed plugin resource
- Easy environment configuration in the user interface
- Supports distributed builds with master-slave architecture
- Build schedules based on expressions
- Supports shells and Windows command execution in pre-build steps
- Supports notification on the build status

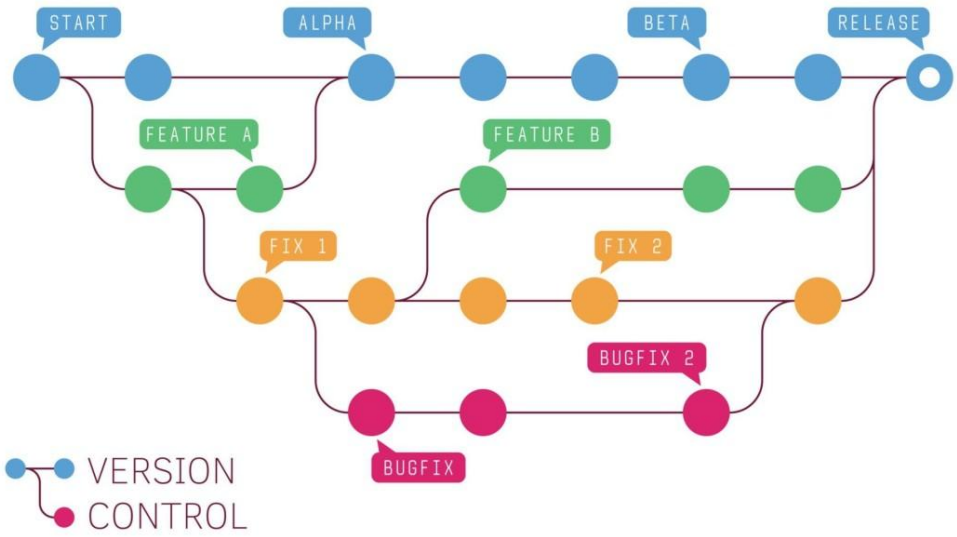
Learning Activity: CI/CD Pipeline using Jenkins

1. To create a new project select the option available in the Dashboard which is **"New Item"**	
2. Now a list of options will be visible on the screen, along with a field to name the pipeline. Add a suitable name and select the **"Pipeline"** option to proceed.	

3. Once redirected, the configuration page will appear.	
4. At first, there is the **General** section where the user can add a description based on the project for which the pipeline has to be created.	
5. Now comes the second section, i.e. **"Build triggers".** Here, we need to specify the branch and repository and give the credentials too.	

6. This is the last section i.e. **“Pipeline”**. Here the user specifies from where the scripts will be imported including the path to the file, repository, credentials, etc.	
7. Now click on **try sample Pipeline** and select **Hello World Script** and click on **Save.**	
8. Now Click on **Build Now** and you will see the Stage View of the Pipeline	

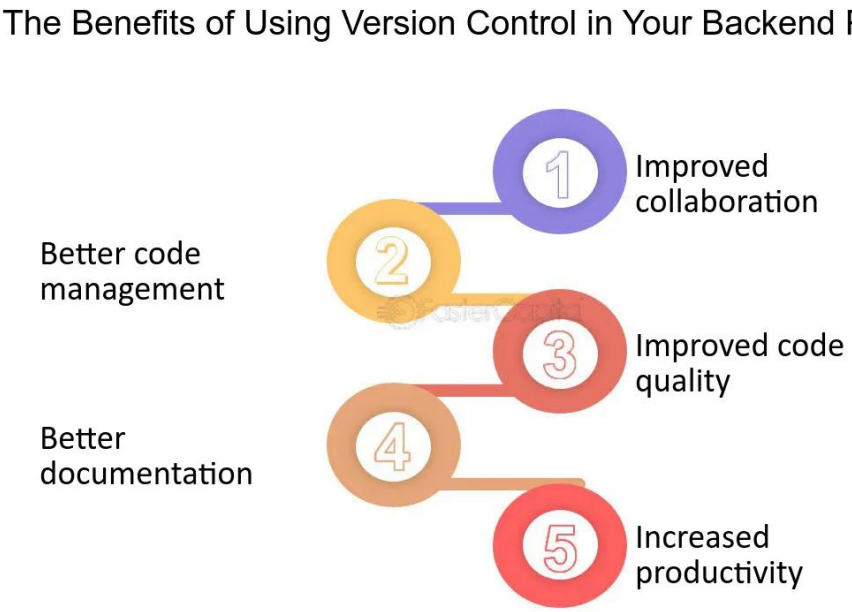
Leveraging Version Control for Collaboration and Consistency



What are Version Control Systems?

VCS are tools that enable developers to manage changes to code over time. This is achieved by tracking each change, or commit, made to the codebase, creating a timeline of the codes evolution. VCS provide a central repository for the code, accessible to all members of the team, enabling them to work on the codebase simultaneously without conflicts. Additionally, VCS enable developers to roll back changes, compare versions of the code, and collaborate on new features.

Benefits of Version Control



Collaborating with Others in Version Control

Module 3: Hands-on CI/CD Pipeline Setup

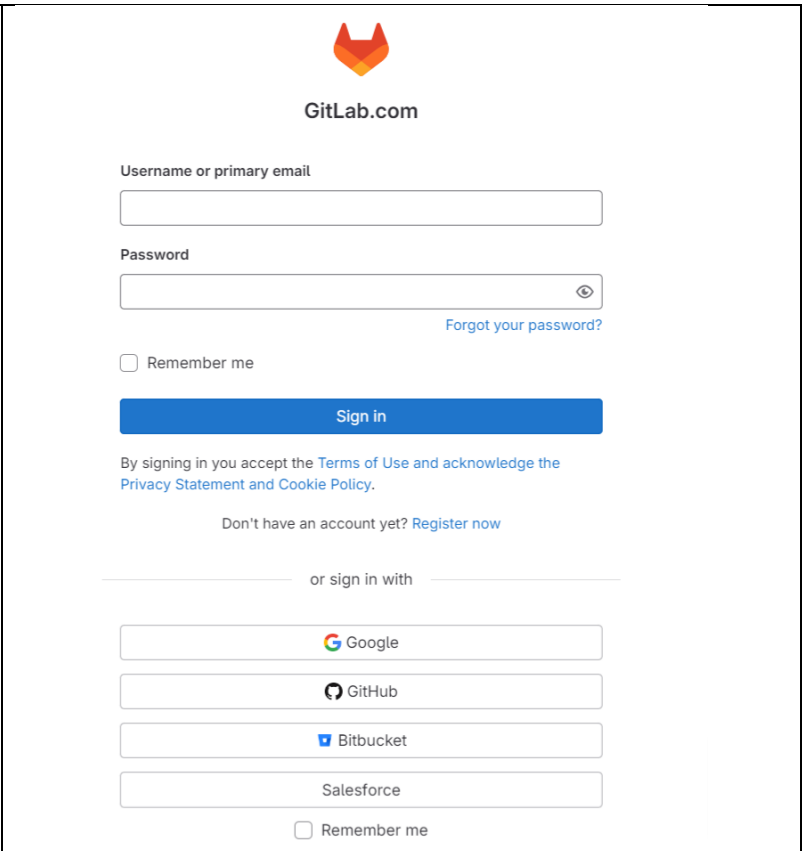
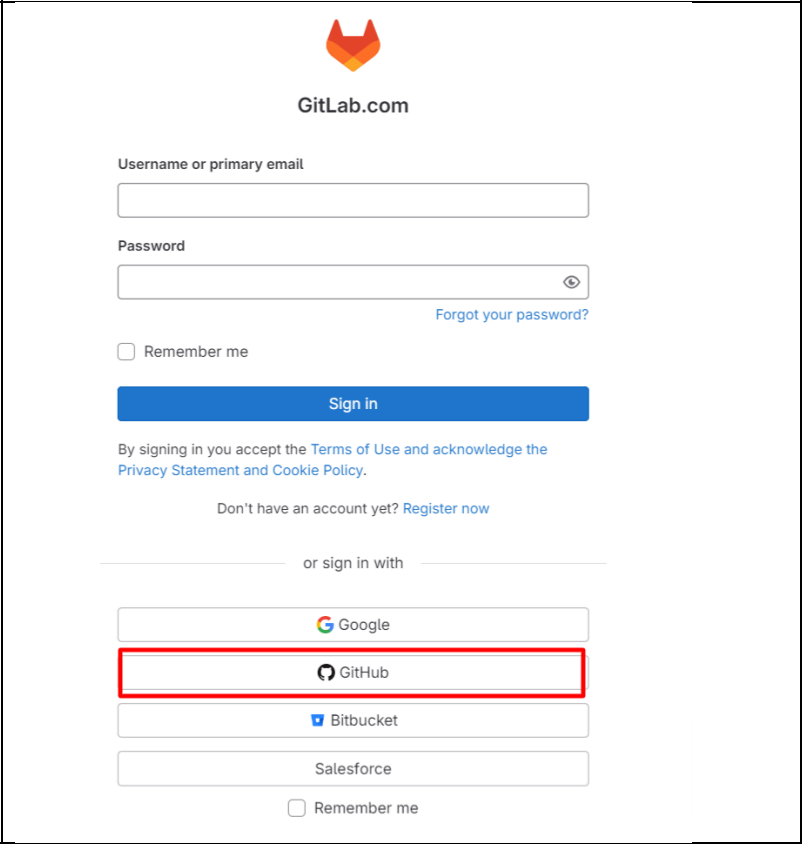
GitLab is a suite of tools for managing different aspects of the software development lifecycle. The core product is a web-based Git repository manager with features such as issue tracking, analytics, and a Wiki.

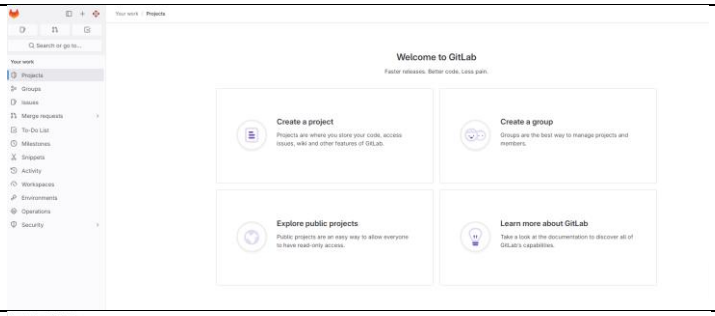
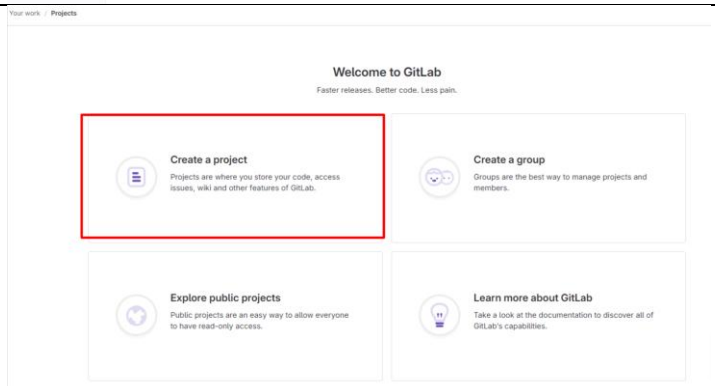
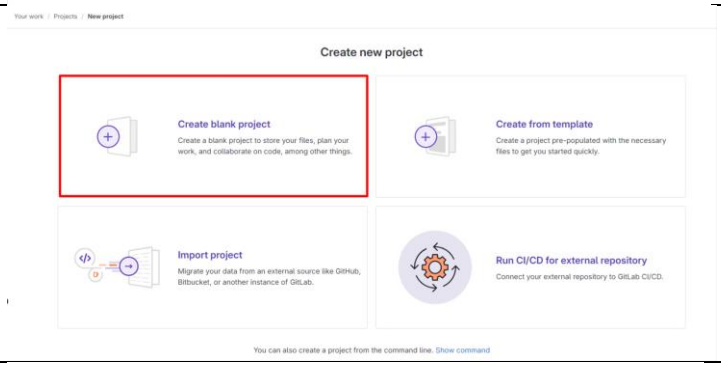
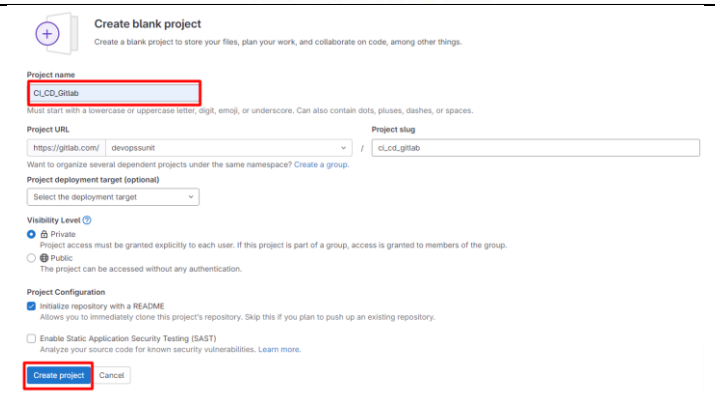
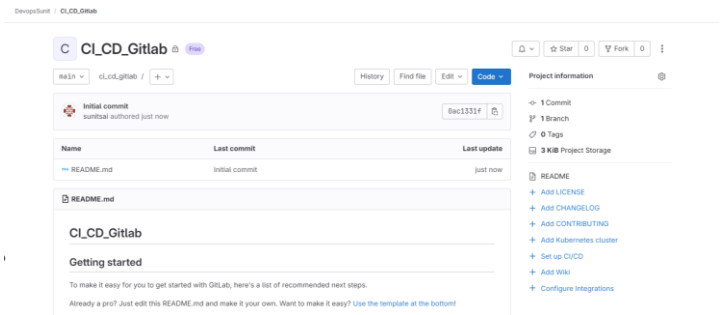
- GitLab allows you to trigger builds, run tests, and deploy code with each commit or push. You can build jobs in a virtual machine, Docker container, or on another server.

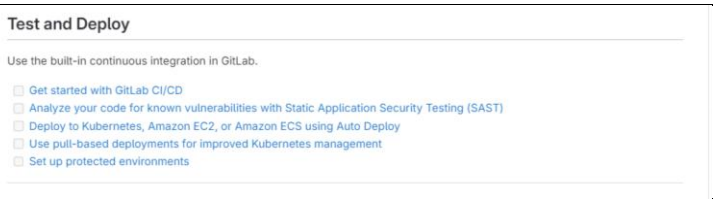
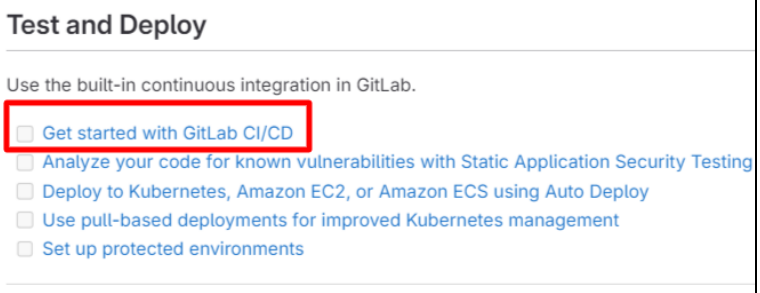
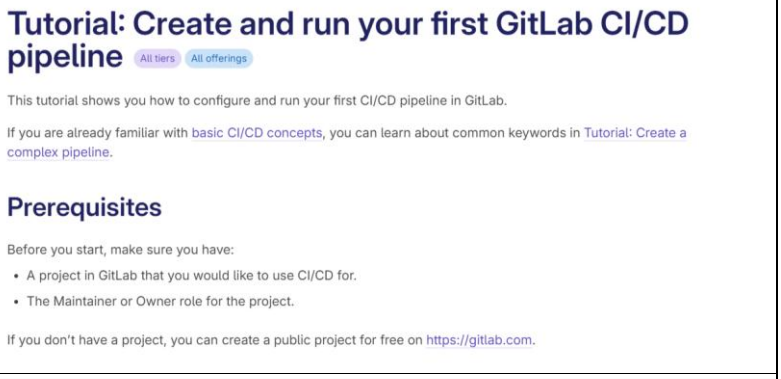
GitLab key features:

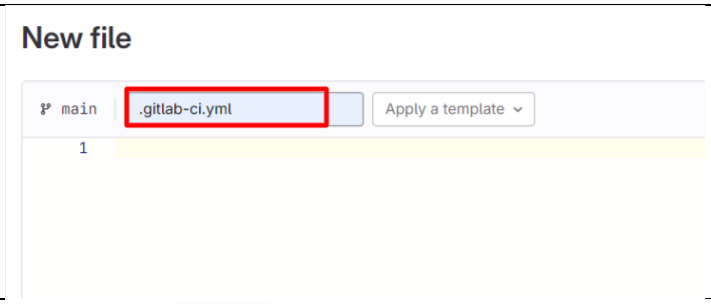
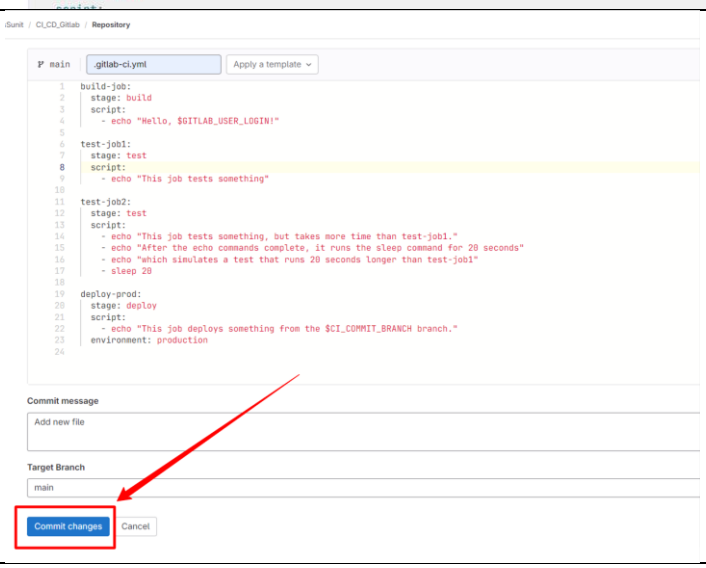
- View, create, and manage codes and project data through branching tools
- Design, develop, and manage codes and project data from a single distributed version control system, enabling rapid iteration and delivery of business values
- Provides a single source of truth and scalability for collaborating on projects and code
- Helps delivery teams fully embrace CI by automating the builds, integration, and verification of source codes
- Provides container scanning, static application security testing (SAST), dynamic application security testing (DAST), and dependency scanning to deliver secure applications along with license compliance
- Helps automate and shorten releases and delivery of applications
- **License**: GitLab is a commercial tool and free package. It offers hosting SaaS on GitLab or on your instance on-premises and/or on the public cloud.

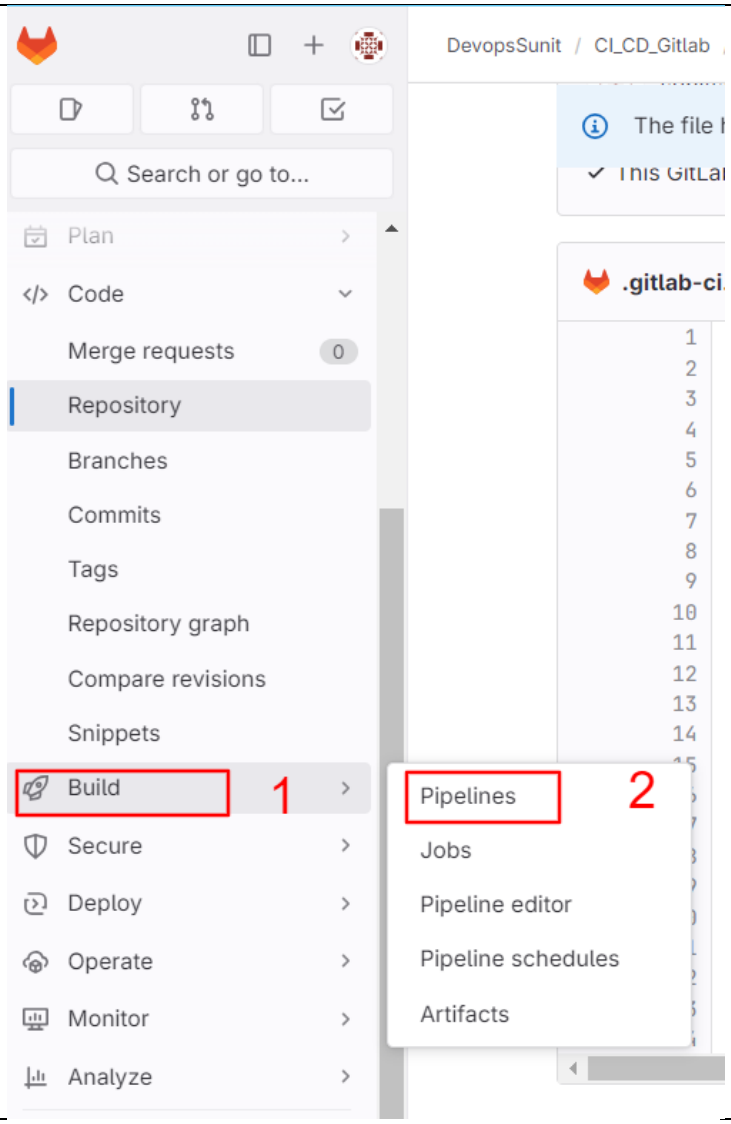
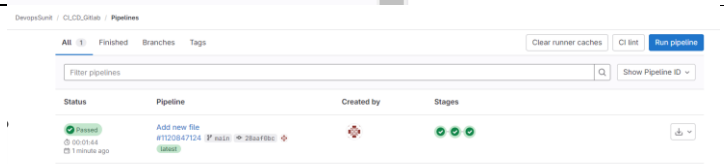
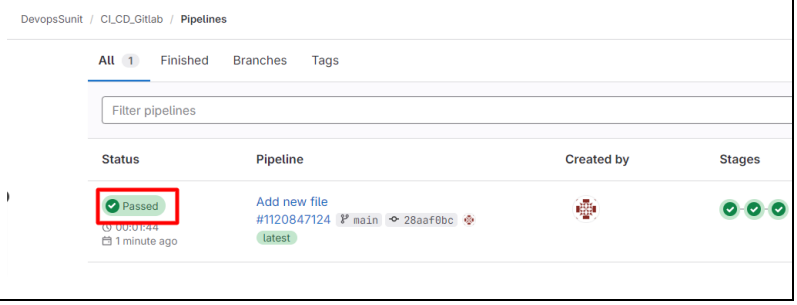
Learning Activity: Setting Up a Basic CI/CD Pipeline

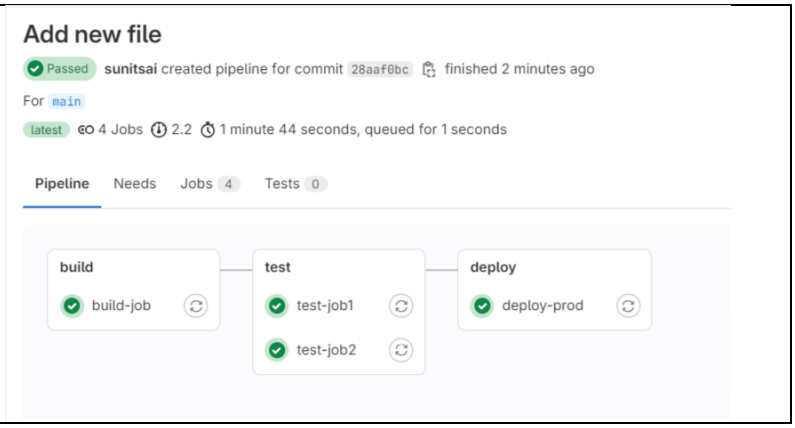
1. Open GitLab the follow link https://gitlab.com/users/sign_in	
2. Login **GitLab** using **GitHub Account.**	

3. After login we are able to see the **GitLab Dashboard**	
4. Now click **Create Project** on dashboard	
5. Now we will see the **Create New Project** Dashboard. 6. Now select the **Create Blank Project**	
7. Now give the Project name show in the figure	
8. Now you will see a project file page shown in figure.	

9. Now scroll down and come on the **Test and Deploy** section on same page.	
10. Now select the **Get Started with GitLab CI/CD** for the helping document. 11. OR 12. You can use this link to open the helping document. https://docs.gitlab.com/ee/ci/quick_start/index.html	
13. Now go to helping document page.	
14. Now again come on Project page on **GitLab** and select **New File.**	

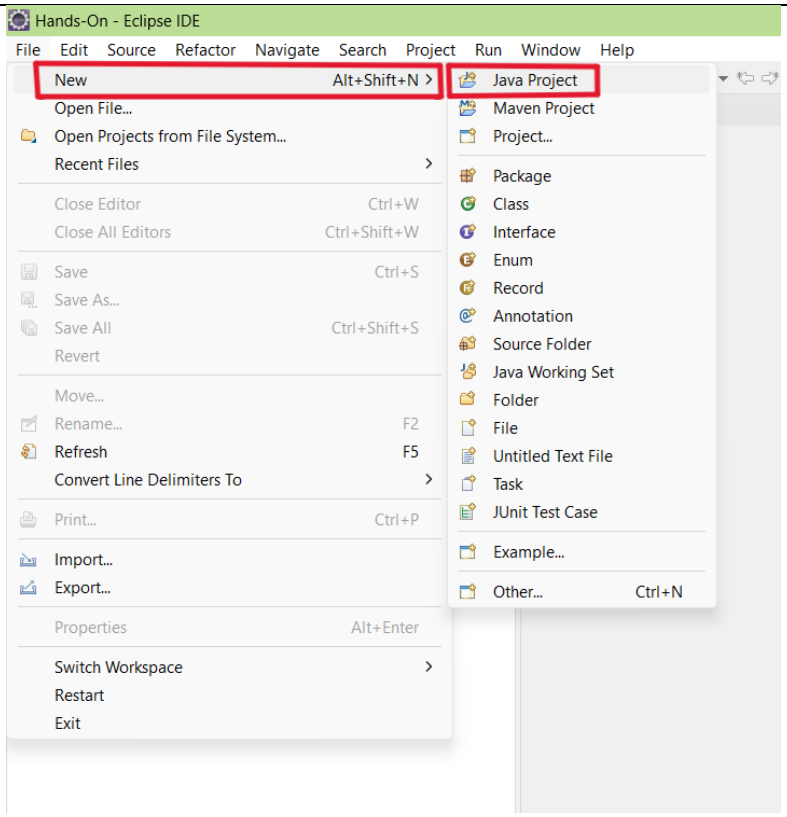
15. Give the file name `.gitlab-ci.yml`	
16. Now open the helping document and scroll down till the **Create a .gitlab-ci.yml file** section and copy the code given in **Step-3**	
17. Paste the code in new created file and click on **Commit Changes.**	

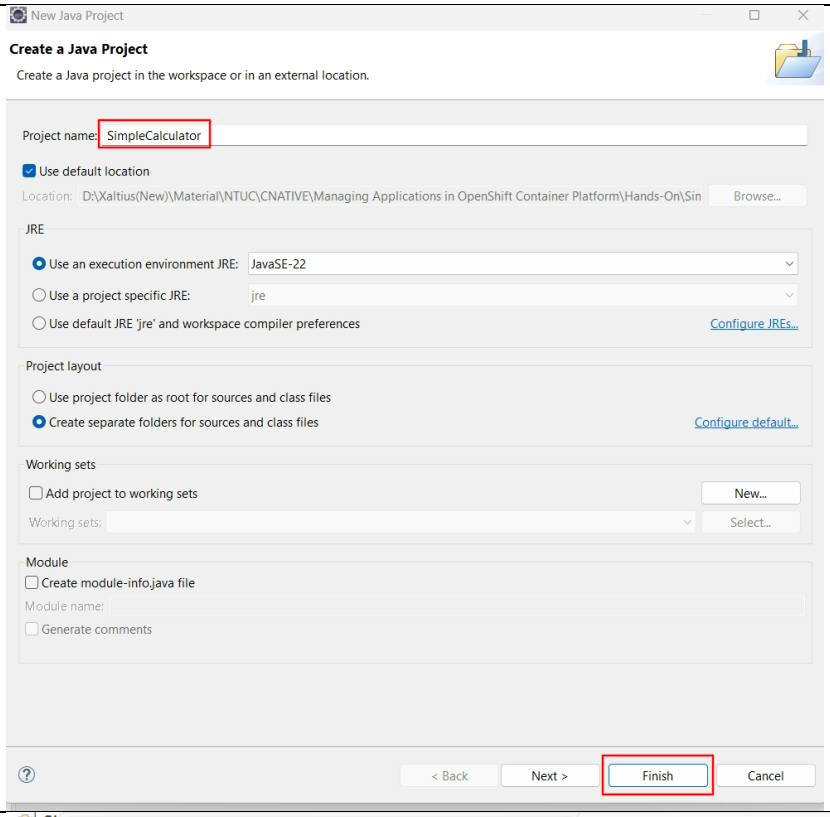
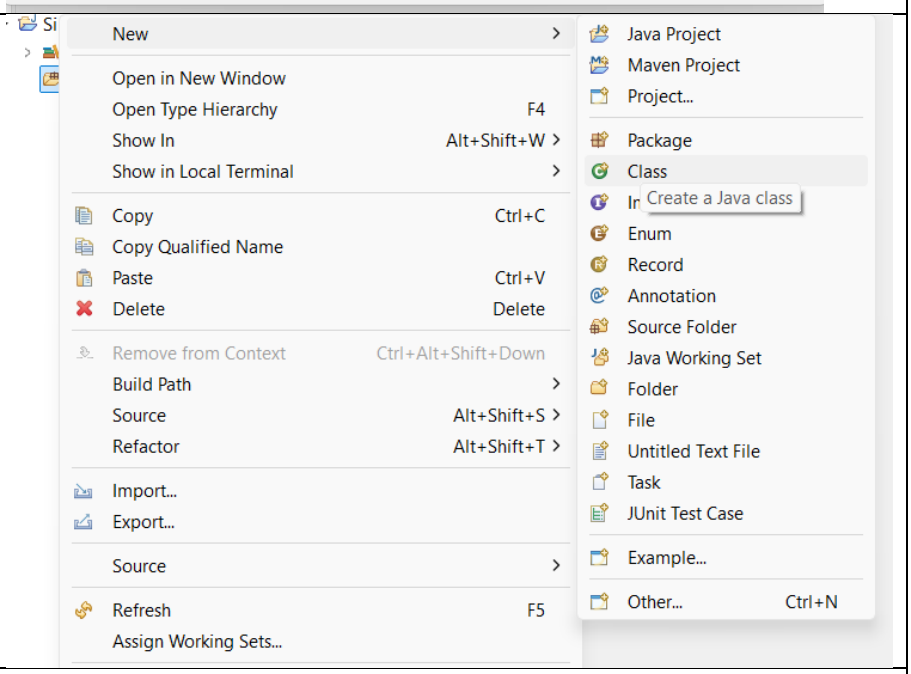
18. Now click on **Build** on side panel and select **Pipeline.**	
19. Now you will see the **Pipeline Page.**	
20. To see the pipeline click on **Status.**	

21. You will see the **Pipeline and Stages.**	

Configuring Automated Testing, Build, and Deployment Stages

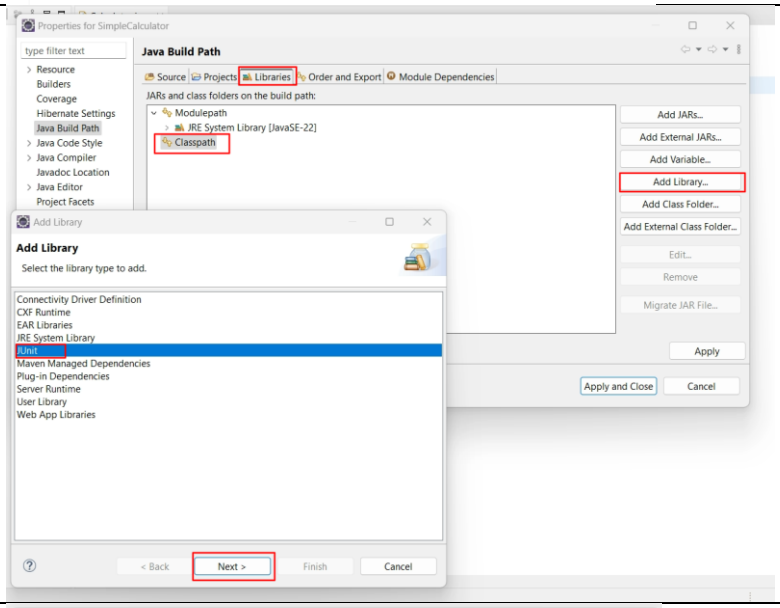
Here's a simple, step-by-step **automated testing** tutorial using **Java** with a basic **JUnit** test. This example will test a simple calculator that can add two numbers.

1. Open Eclipse IDE and click on File→ New → Java Project	

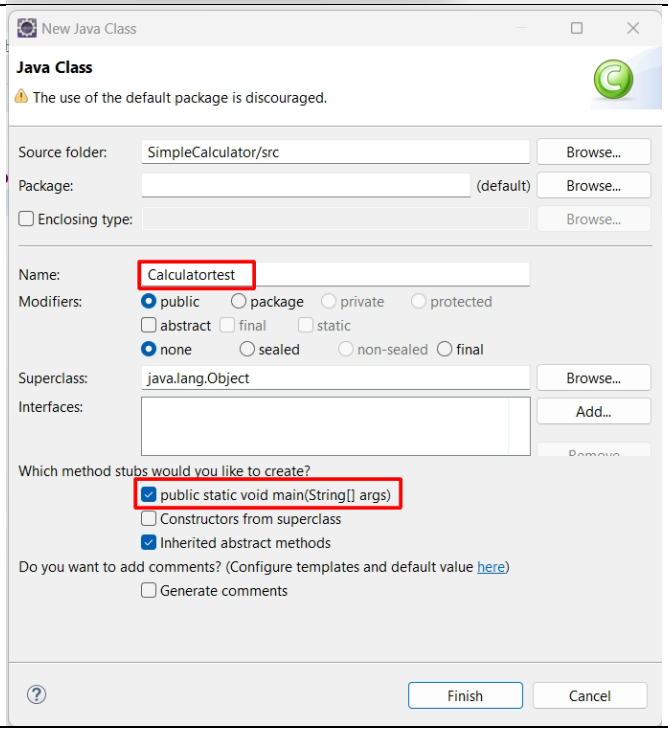
2. Create a Project: • Enter a name for your project, e.g., SimpleCalculator. • Click **Finish**.	
3. Create a New Class: • Right-click on the src folder in the Package Explorer on the left panel. • Select New → Class. • In the dialog box, name the class Calculator and click Finish.	
4. Now add the following code to the Calculator.java file.	``` public class Calculator { public int add(int a, int b) { return a + b; } } ```

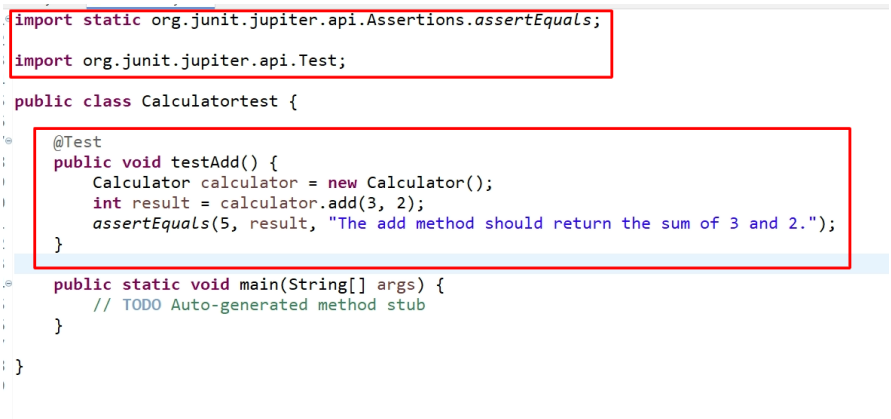
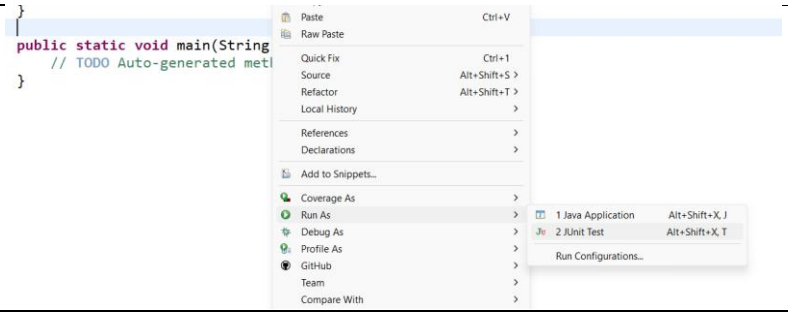
```java
public class Calculator {
    public int add(int a, int b) {
        return a+b;
    }
}
```

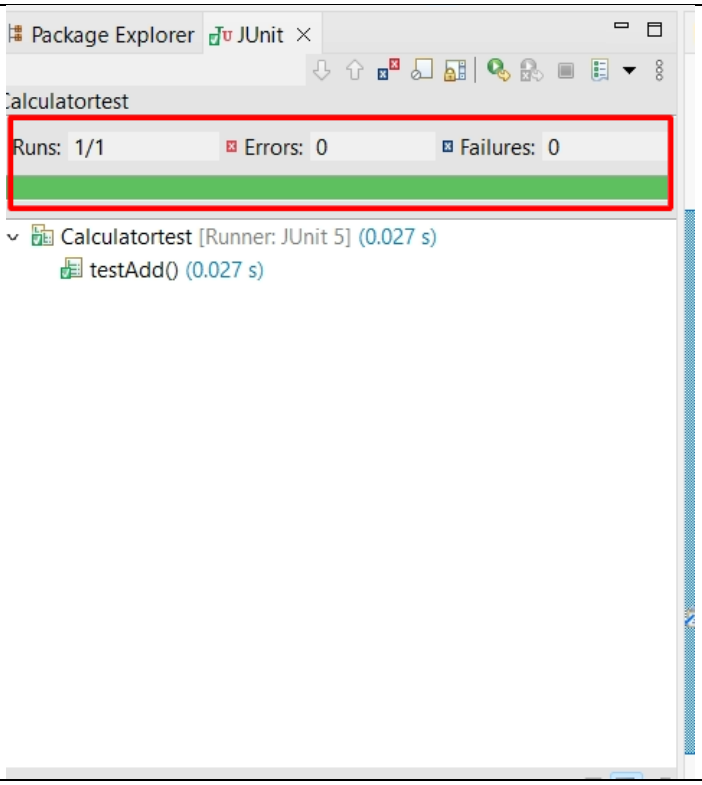
5. Add JUnit Library:
   - Right-click on your project in the Package Explorer.
   - Select Build Path → Configure Build Path.
   - In the Libraries tab, click Add Library → Select JUnit → Click Next → Choose JUnit 5 and click Finish.



6. Create a New Test Class:
   - Right-click on the src folder again.
   - Select New → Class and name it CalculatorTest.java
   - Click the checkbox labeled public static void main(String[] args) to automatically generate the main method for your test class.



7. Now add the following test code to the CalculatorTest.java file to test if the add method works correctly.	import static org.junit.jupiter.api.Assertions.assertEquals; import org.junit.jupiter.api.Test; public class CalculatorTest { @Test public void testAdd() { Calculator calculator = new Calculator(); int result = calculator.add(3, 2); assertEquals(5, result, "The add method should return the sum of 3 and 2."); } } 
8. Run the Test: • Right-click on CalculatorTest.java in the Package Explorer. • Select Run As → JUnit Test.	

9. Understanding the Test Results • **Green Bar:** This means the test passed successfully, and the add method returned the correct result. • **Red Bar:** If the bar is red, the test failed. This means the add method did not return the expected result, and the problem needs to be fixed.	

Implementing Version-controlled Infrastructure as Code

Terraform will prompt for confirmation before destroying the resources. This is useful for cleanup or testing purposes.

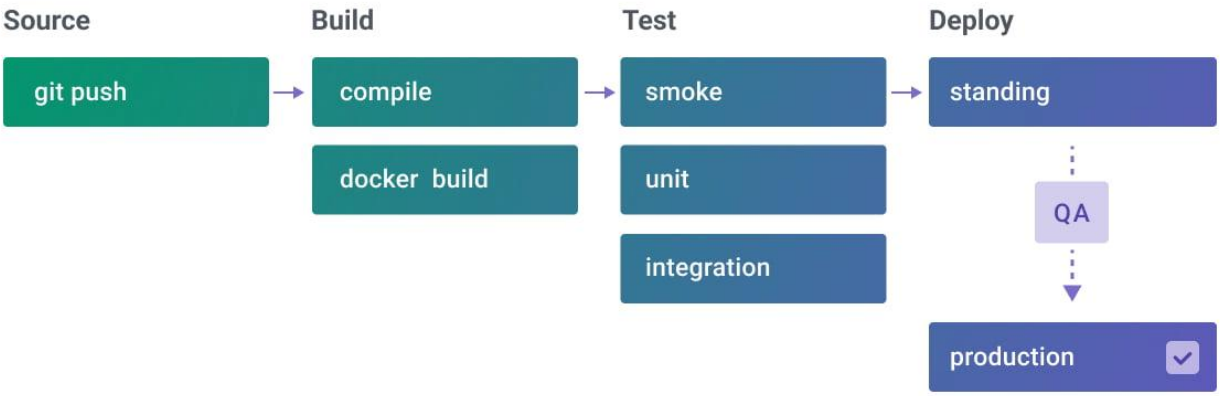
Key Concepts for Version-Controlled IaC

1. **Configuration Files**: Store all infrastructure configurations in files (.tf files for Terraform).
2. **State Management**: Terraform keeps track of your infrastructure state in a terraform.tfstate file. This file is important for versioning and tracking the state of your infrastructure.
3. **Version Control**: Store configuration files in a Git repository to track every change.
4. **Automation**: Use CI/CD pipelines with Terraform to automate infrastructure deployment based on the latest code changes in your Git repository.

Advanced Considerations

- **Remote State Storage**: Store the Terraform state file in a remote location (like AWS S3 or Azure Blob Storage) to collaborate on infrastructure management with teams.
- **Modules**: Organize your Terraform code into reusable modules for complex infrastructure setups.
- **CI/CD Integration**: Integrate Terraform with CI/CD tools like Jenkins, GitHub Actions, or GitLab CI to automate deployments from version-controlled IaC code.

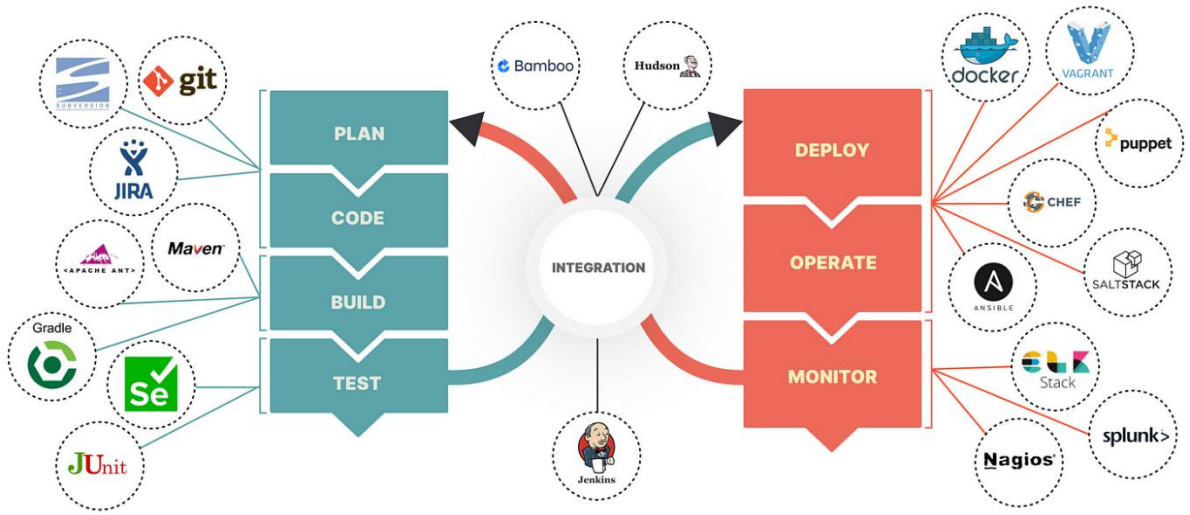
Best Practices for Designing Reliable and Efficient Pipelines



Best Practices

- **Version Control:** Use a robust version control system like Git to manage code changes.
- **Automated Testing:** Write unit tests, integration tests, and end-to-end tests to validate code changes automatically.
- **Automated Builds:** Set up automated build pipelines that compile code, package applications, and create artifacts.
- **Frequent Integration:** Encourage developers to integrate code changes into the shared repository multiple times a day.
- **Feedback Loops:** Implement feedback mechanisms to notify developers of build and test results.

Module 4: Tools and Practices for Continuous Monitoring



Understanding the Importance of Continuous Monitoring

Continuous quality monitoring in DevOps is the process of identifying threats to the security and compliance rules of a software development cycle and architecture. Also known as continuous control monitoring or CCM, this is an automated procedure that can be extended to detect similar inconsistencies in IT infrastructures. Continuous monitoring helps business and technical teams determine and interpret analytics to solve crucial issues, as mentioned above, instantaneously. Digital experience monitoring, or DEM, on the other hand, is the process of optimizing the operational behaviour and experience of a system.

Importance of Continuous Monitoring

Continuous monitoring in DevOps has a few specific goals, as are mentioned below:

1. Many components of software operations can trigger devastating outcomes like breaches. Continuous monitoring aims to strengthen the transparency of such environments while keeping in place a vigilant system to monitor and resolve said issues.
2. Continuous monitoring aims to identify performance inconsistencies and error sources. It also resolves these problems using relevant solutions to safeguard the enterprise.
3. Continuous monitoring assists companies in keeping a tab of their user experience. CM is especially helpful in tracking user feedback after a recent change or update to a software or an application. This helps firms build and strengthen their business strategies.

DevOps tools for Infrastructure Monitoring

- Nagios
- Prometheus

DevOps Tools for Application Monitoring

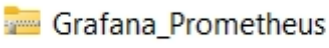
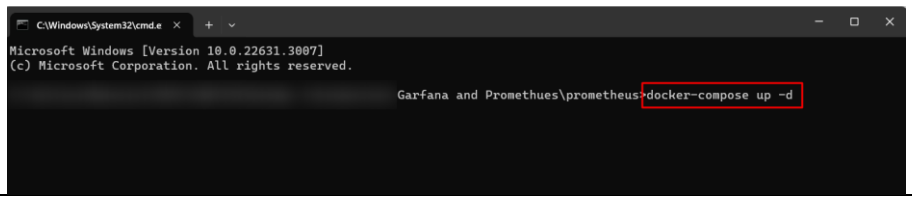
- Datadog
- Splunk

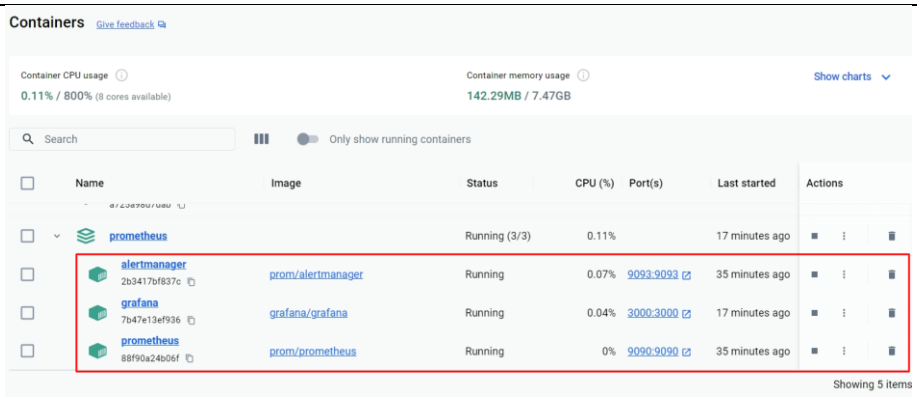
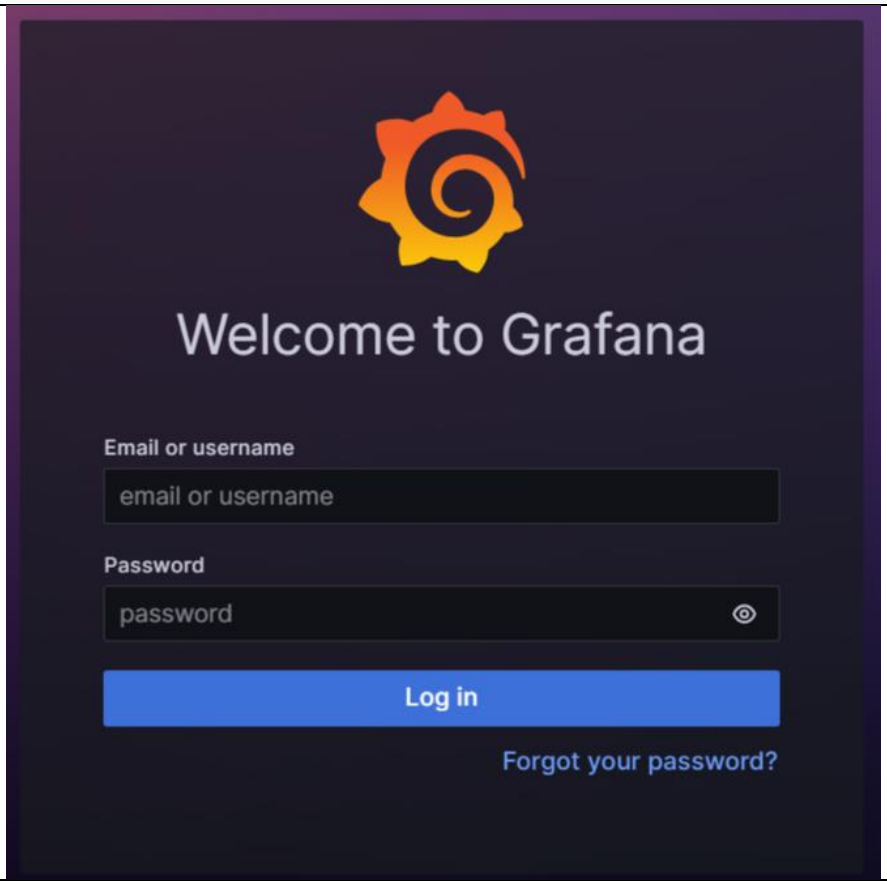
DevOps Tools for Networking Monitoring

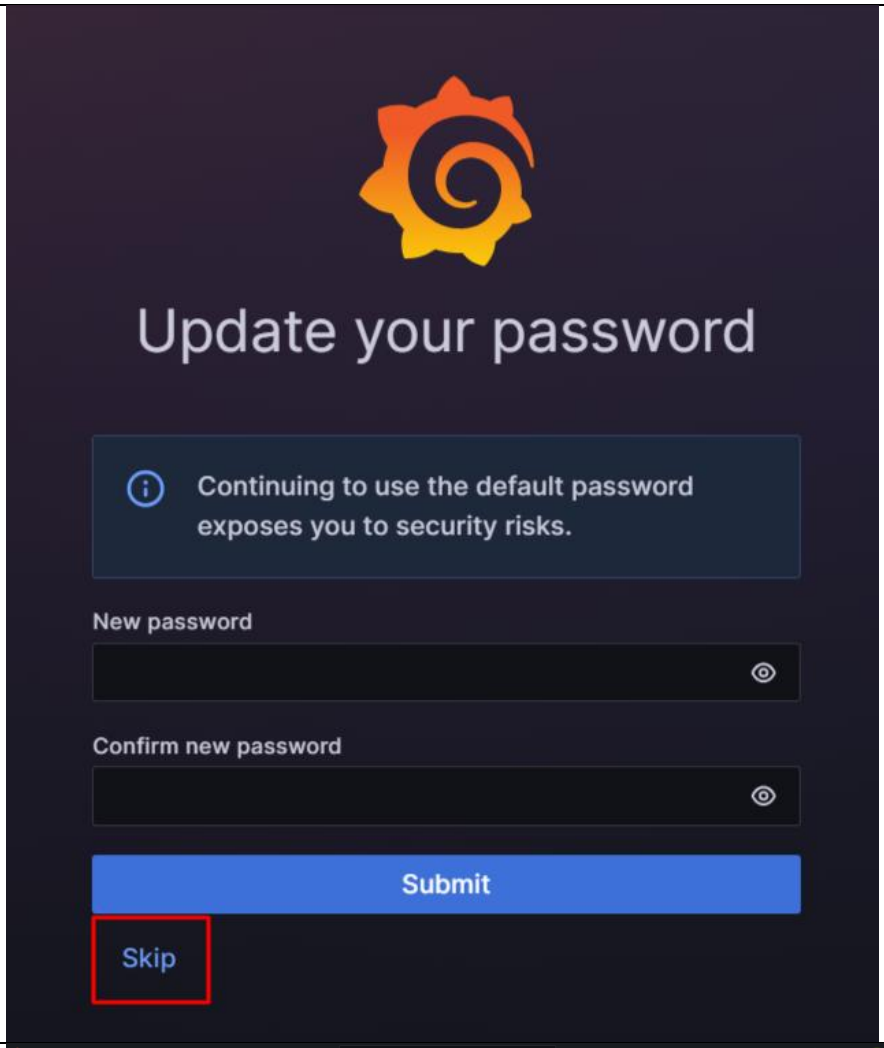
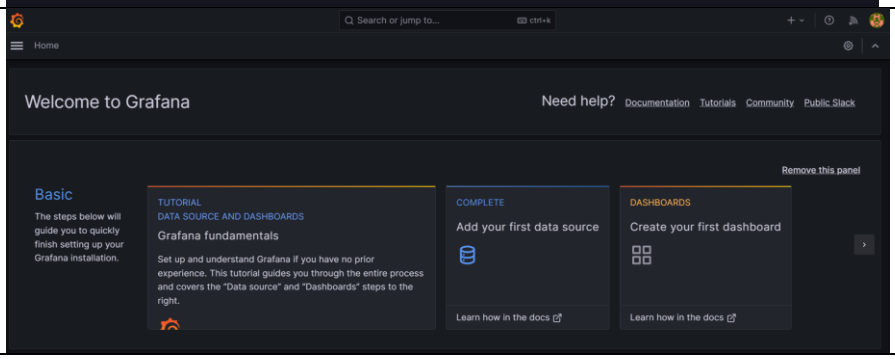
- Wireshark
- NMap- Network Mapper

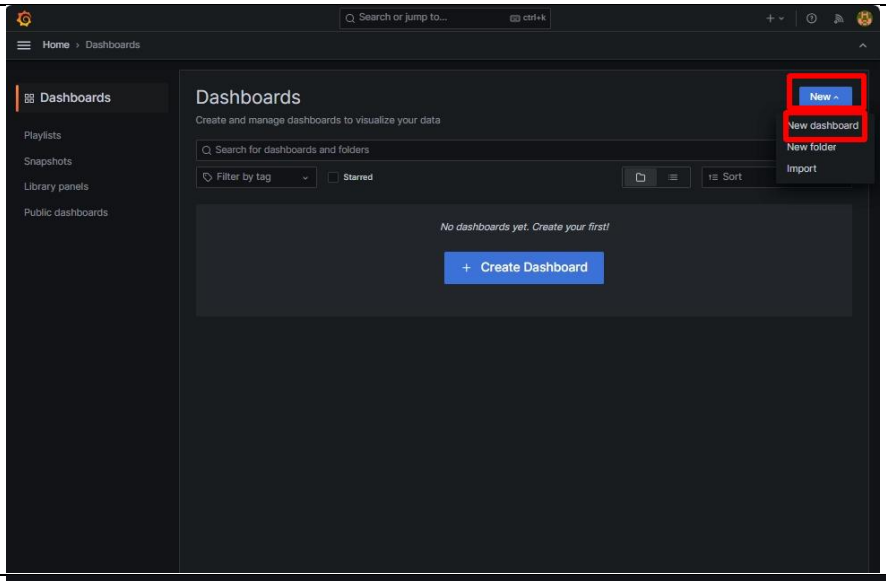
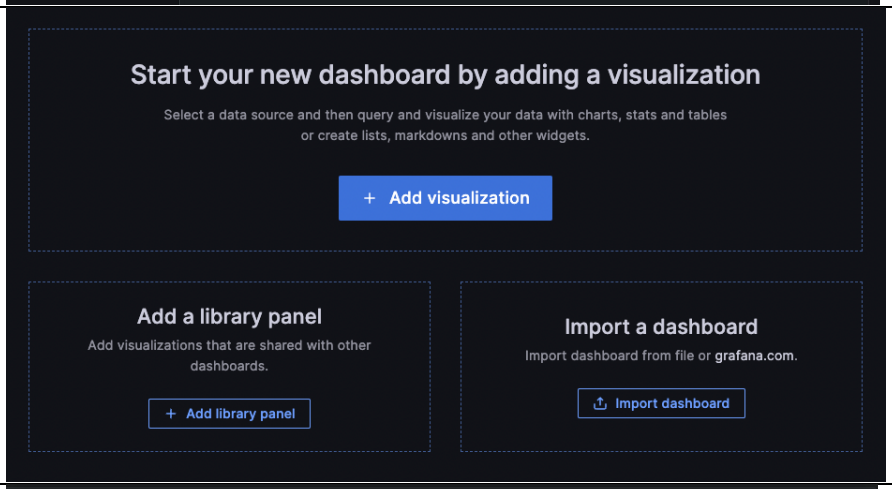
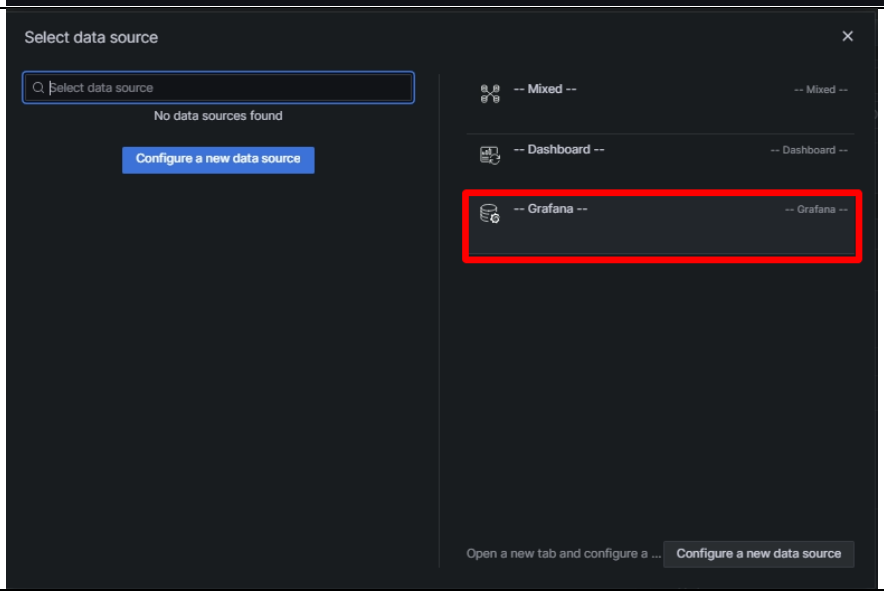
Exploring Monitoring Tools: Prometheus, Grafana, ELK Stack

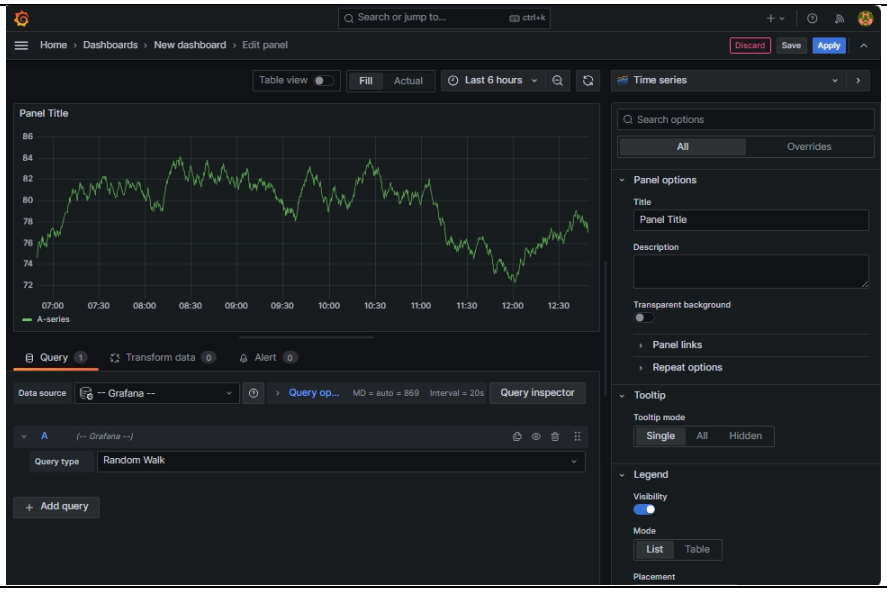
Setup the Grafana Monitoring Tool:

1. Download the **Zip file** named **Grafana_Prometheus** from the **Example** folder**.**	
2. Extract that file and go inside that file and open CMD.	
3. Now type the following command in CMD to compose the data in the your docker. **4. docker-compose up -d**	

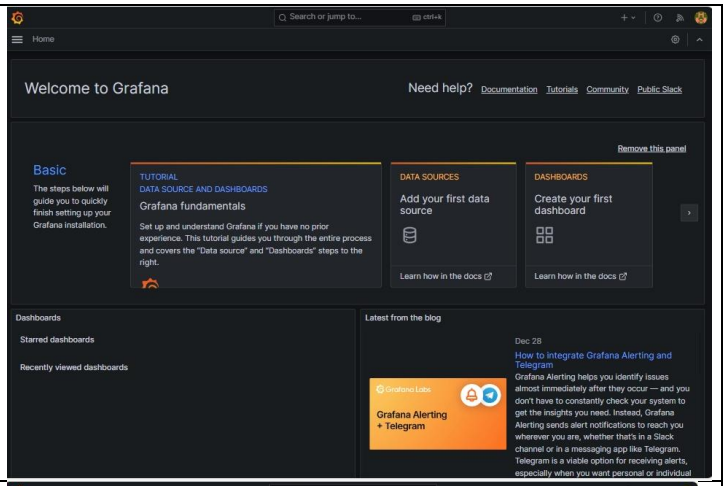
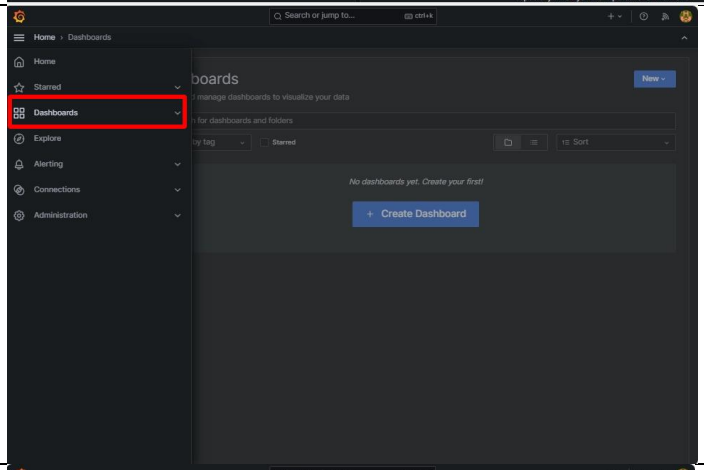
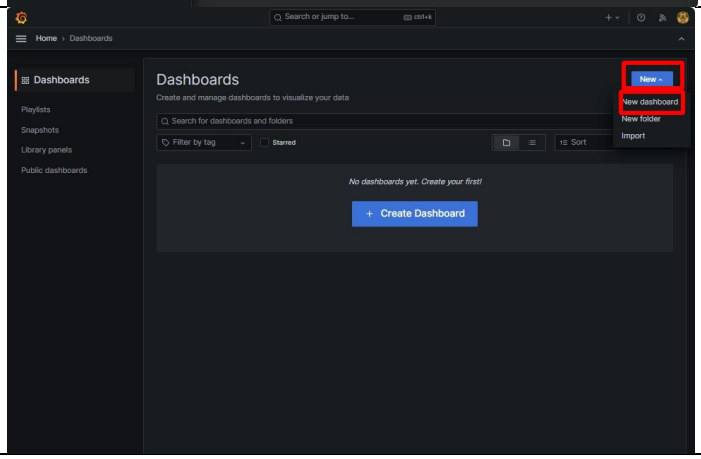
5. Open your docker desktop and you will see the 3 containers for **Grafana**	
6. Now open your browser and run the follow URL to launch the Grafana. 7. http://localhost:3000/login	

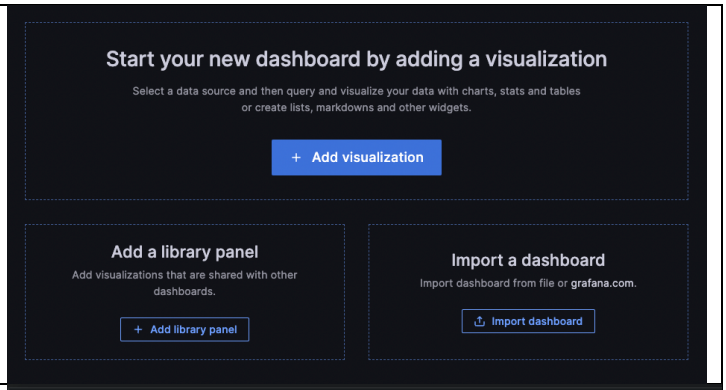
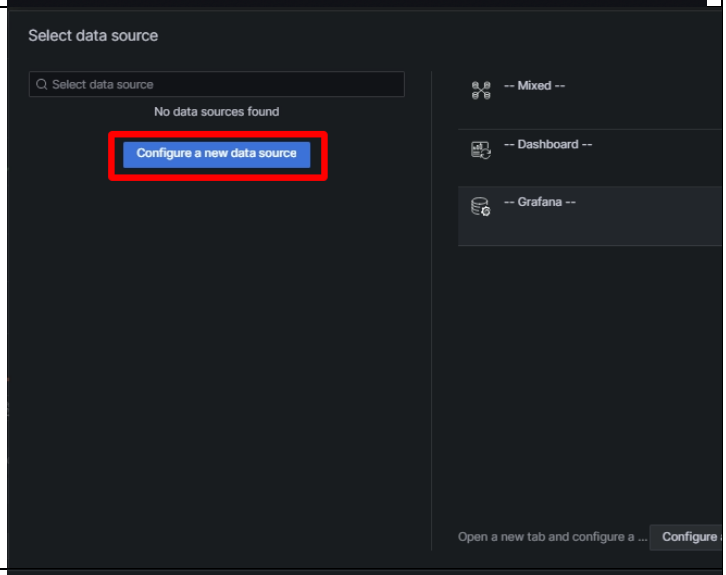
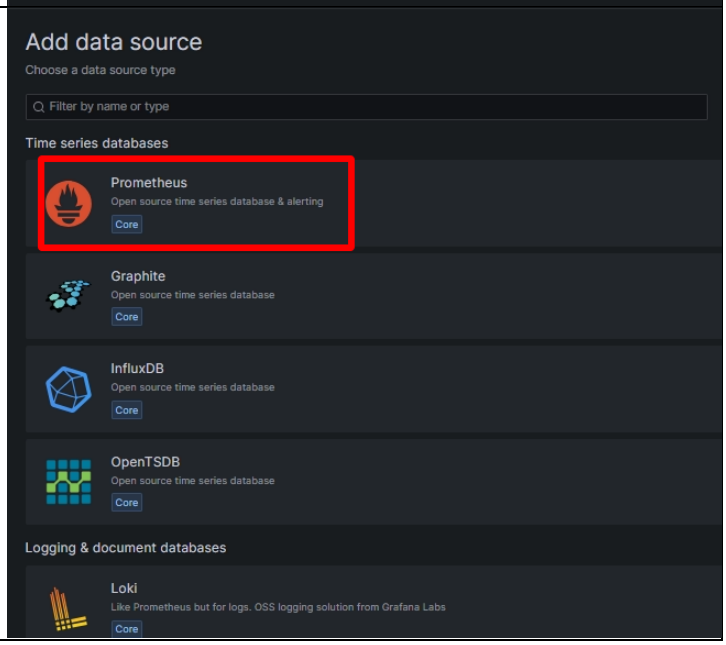
8. Now type the Username and Password. 9. Username: admin 10. Password: admin	
11. You will see the Grafana Dashboard	

12. On the Dashboards page, click **New** and select **New Dashboard** from the drop-down menu.	
13. On the dashboard, click **+ Add visualization**.	
14. In the dialog box that opens, click `-- Grafana --`:	

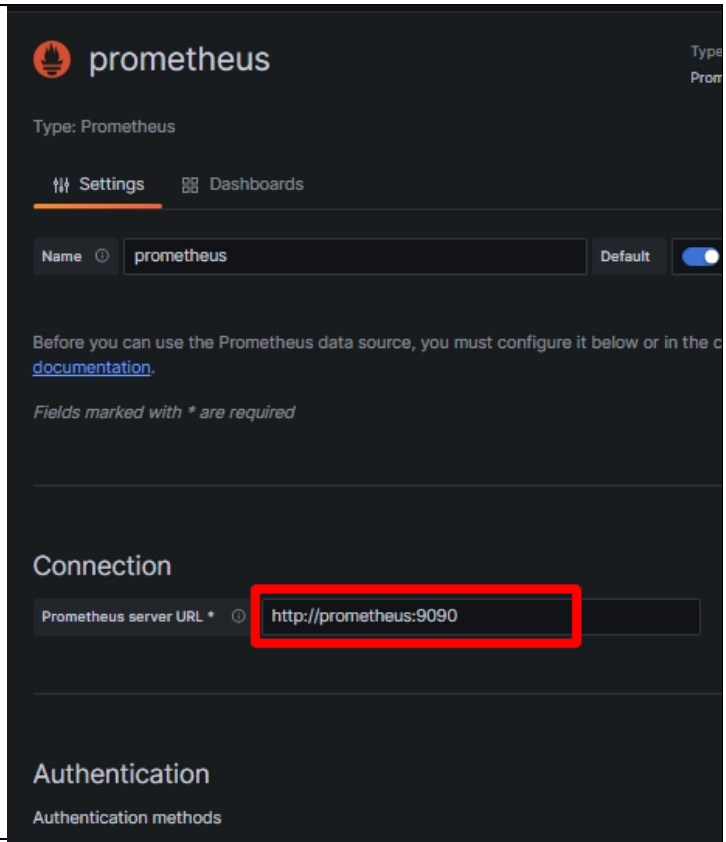
15. Grafana Dashboard	

Setup the Prometheus Monitoring Tool:

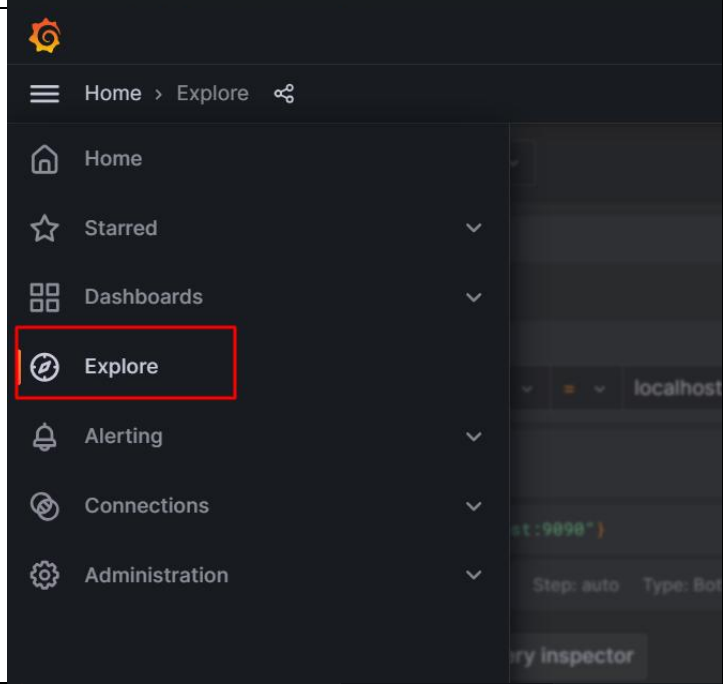
1. Grafana Dashboard	
2. Click **Dashboards** in the left-side menu.	
3. On the Dashboards page, click **New** and select **New Dashboard** from the drop-down menu.	

4. On the dashboard, click **+ Add visualization**.	
5. Now click on **Configure a new data source**	
6. Now click **Prometheus**	

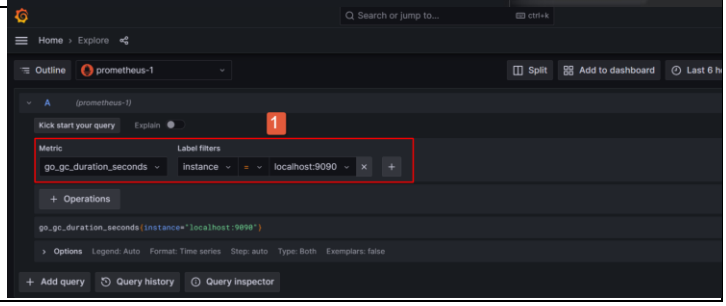
7. Now write down the **Connection URL**
8. **http://prometheus:9090**



9. Now click on **Explore** from **Grafana Side Panel**



10. Now **Prometheus Dashboard** will open.

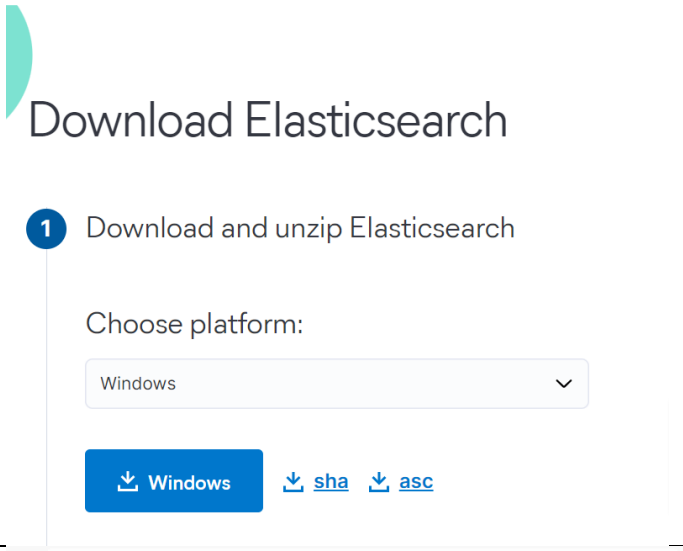
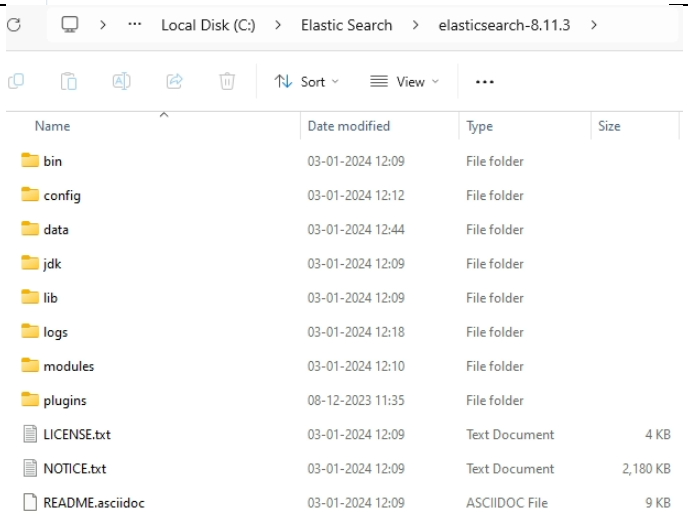


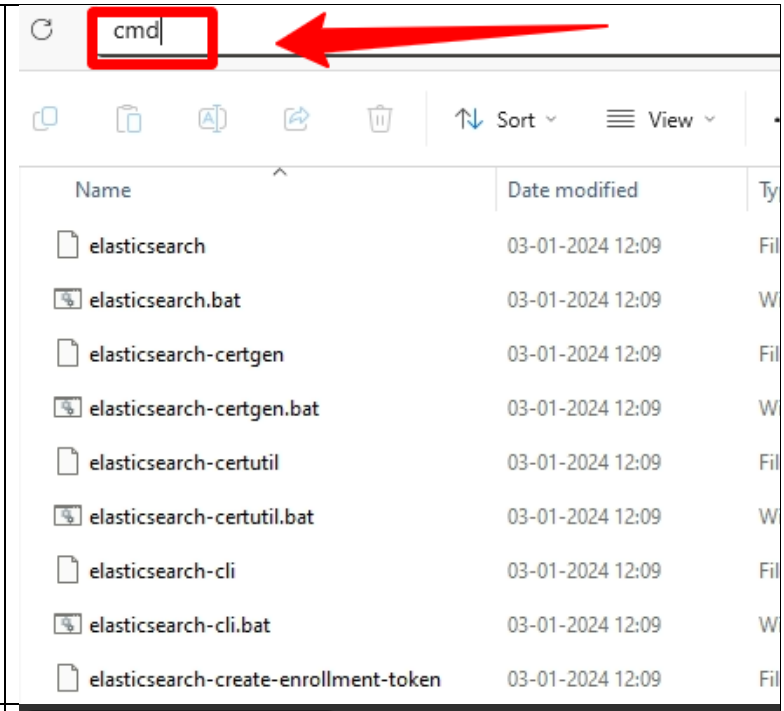
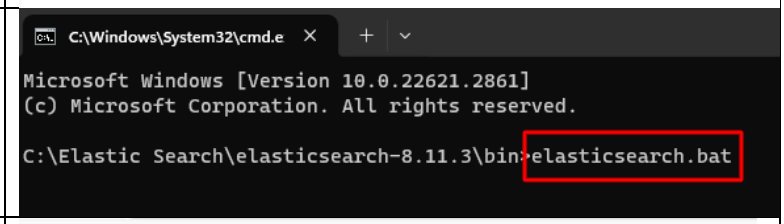
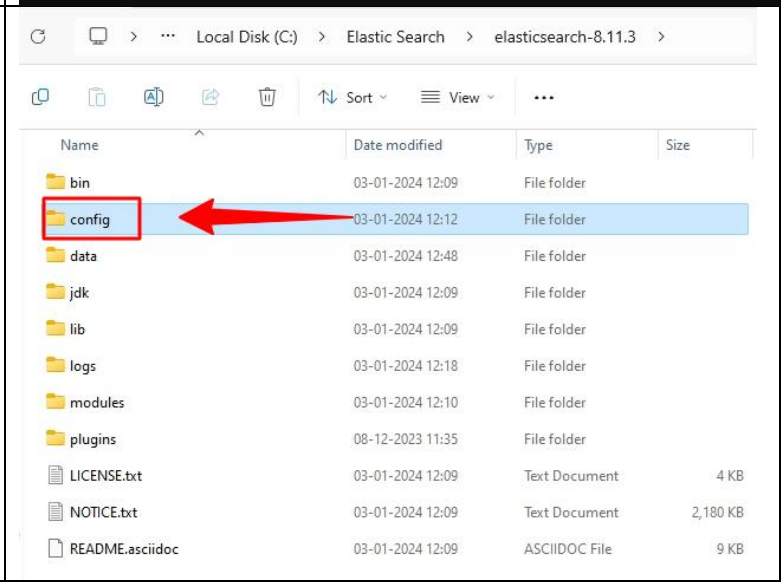
What is the ELK Stack?

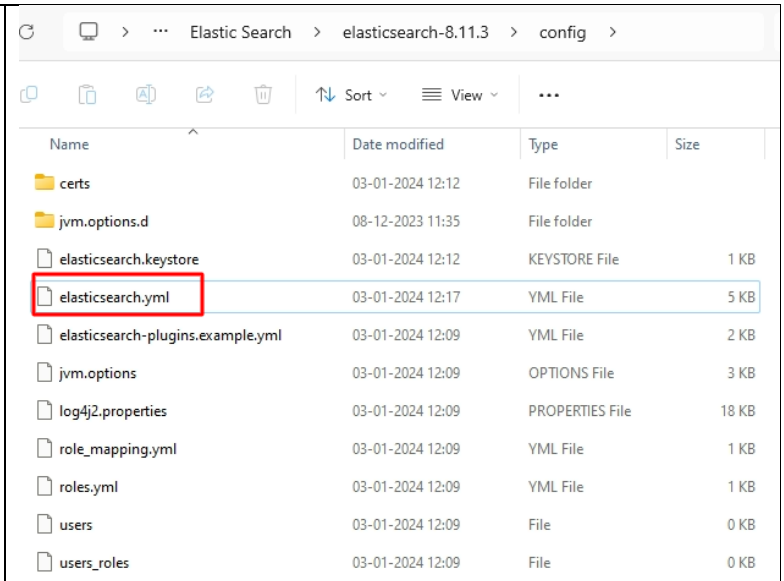
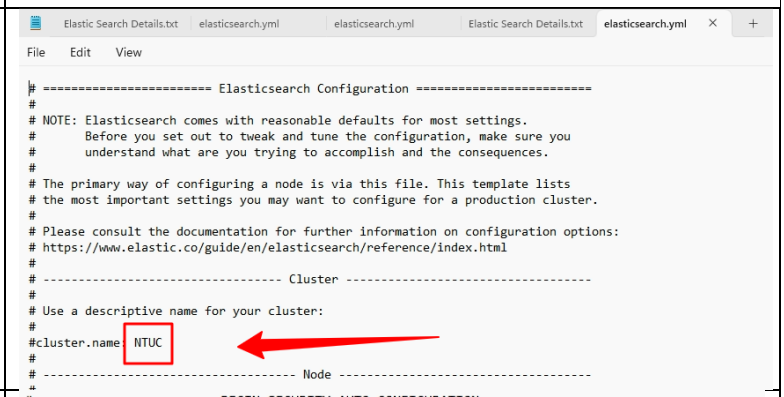
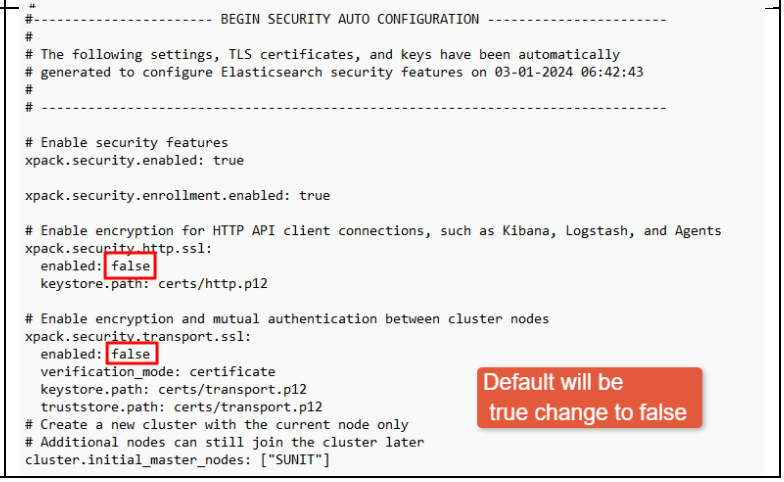
The ELK Stack began as a collection of three open-source products — Elasticsearch, Logstash, and Kibana — all developed, managed and maintained by Elastic. The introduction and subsequent addition of Beats turned the stack into a four-legged project.

How to use the ELK Stack?

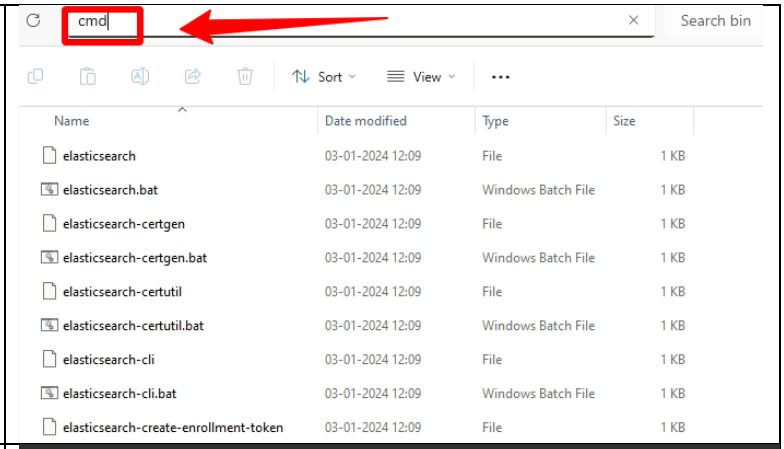
Setup the ELK Monitoring Tool:

1. Download the Elastic Search ZIP from the following link: 2. https://www.elastic.co/downloads/ elasticsearch	
3. Unzip the folder in any of the drives.	

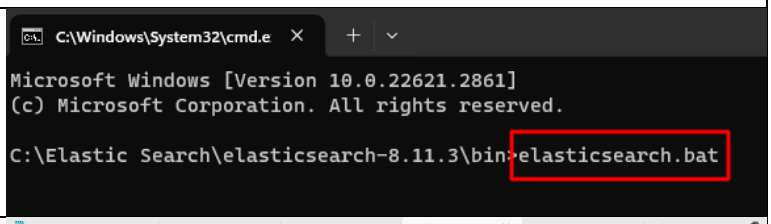
4. Now go in side the **bin folder** and launch the **CMD** with the path of **bin folder.**	
5. Now cmd will open and type **elasticseacrh.bat**	
6. Now go in the **config folder** inside the elastic search root folder.	

7. Open the **elastic search.yml file** in notepad.	
8. Now after open the file change the name of **cluster.name = NTUC**	
9. Scroll down and go to the **Security Section** in Notepad and change the things shown in the figure from **true to false.**	

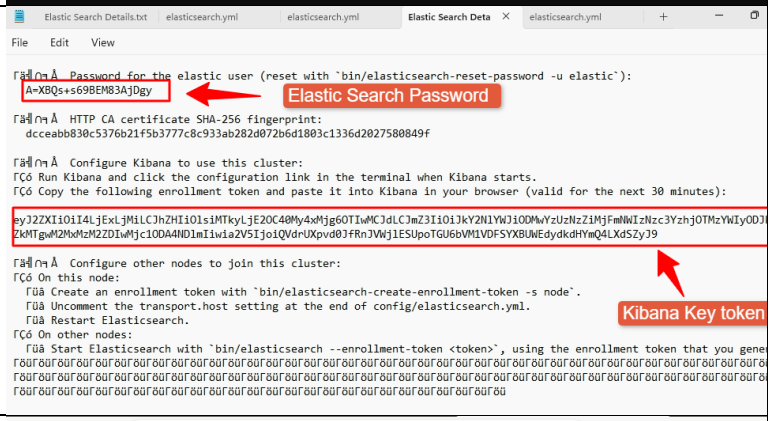
10. Now go in side the **bin folder** and launch the **CMD** with the path of **bin folder.**



11. Now cmd will open and type **elasticseacrh.bat**

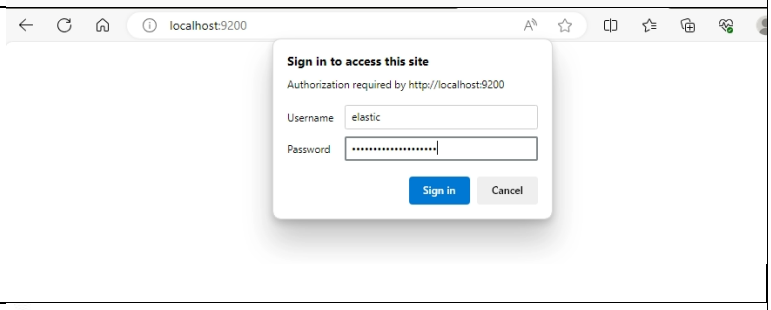


12. Now you will get the password and key token for Kibana copy that and paste it into notepad.



13. Open the browser and type the follow URL:
14. **Localhost:9200**

15. **Username:** elastic
16. **Password:** Copy from notepad which you have save from CMD

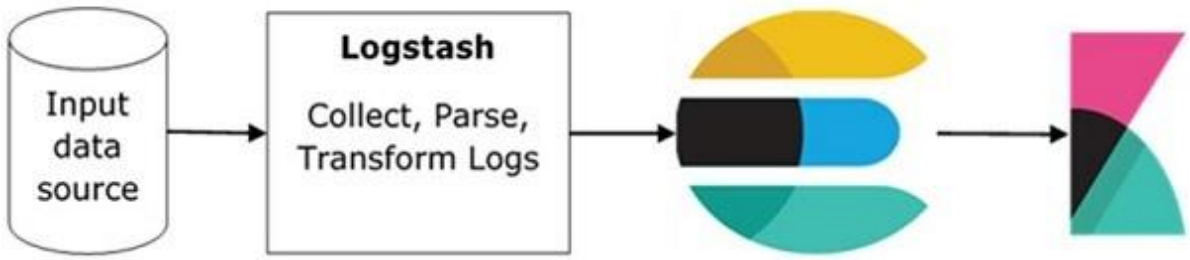


17. **Output**

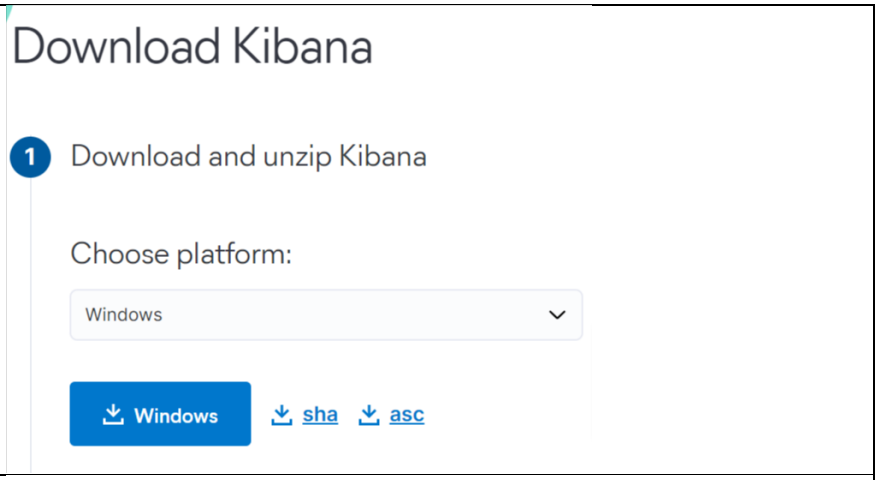
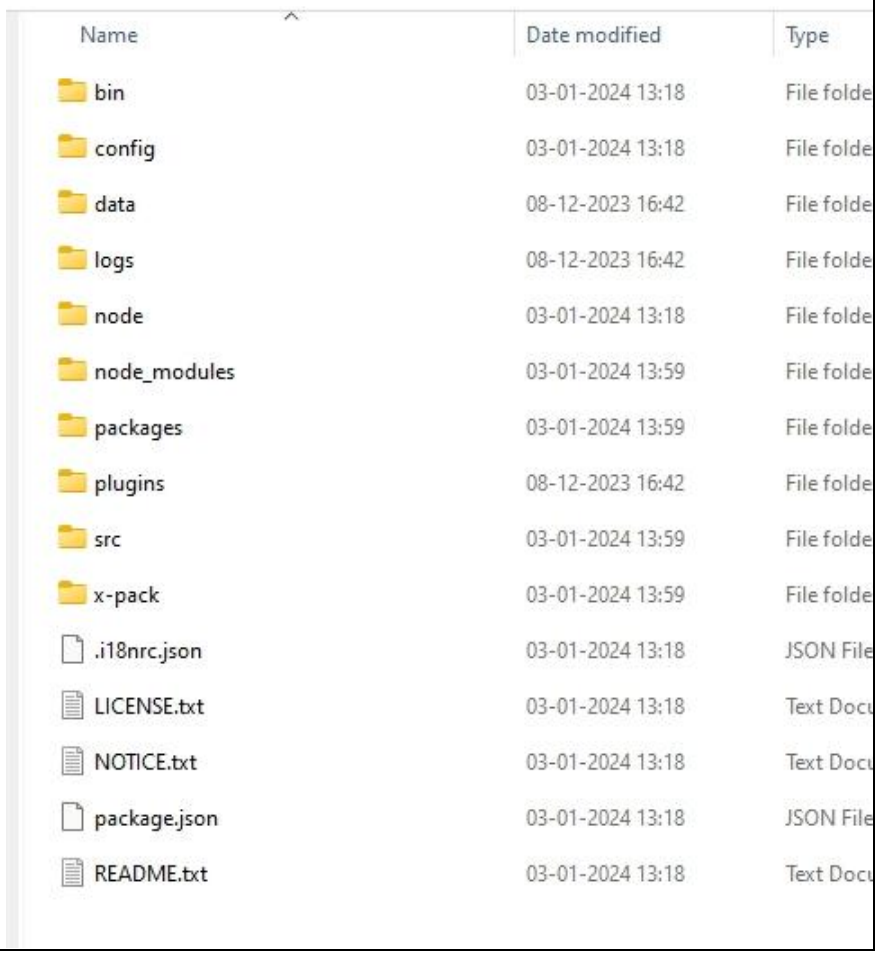
Kibana- Introduction

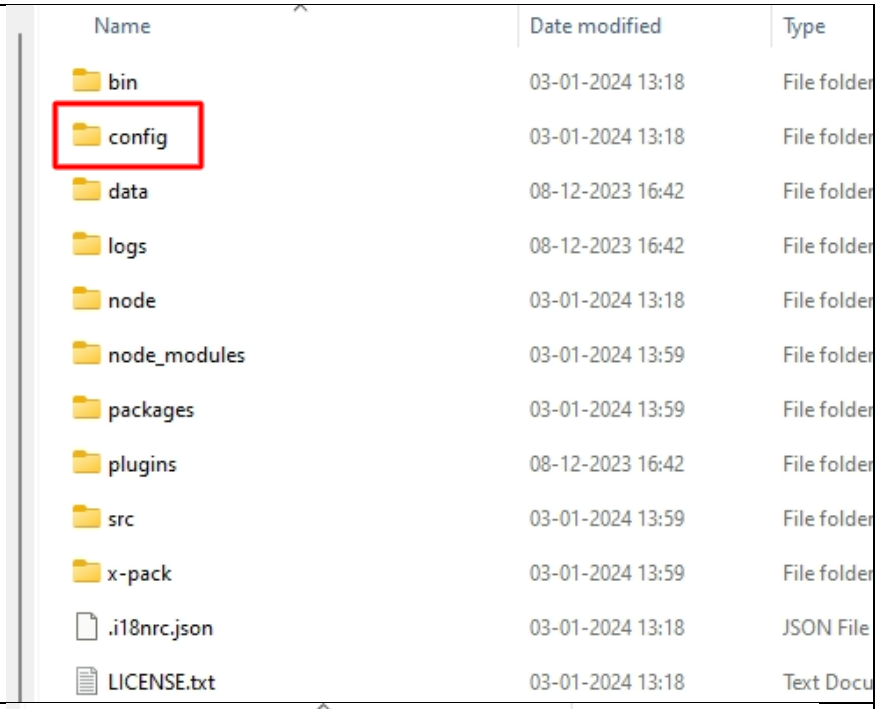
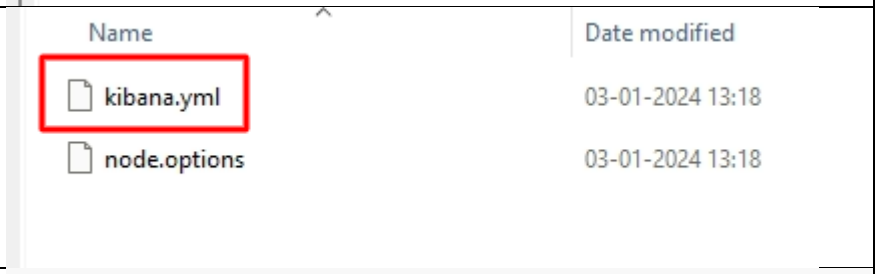
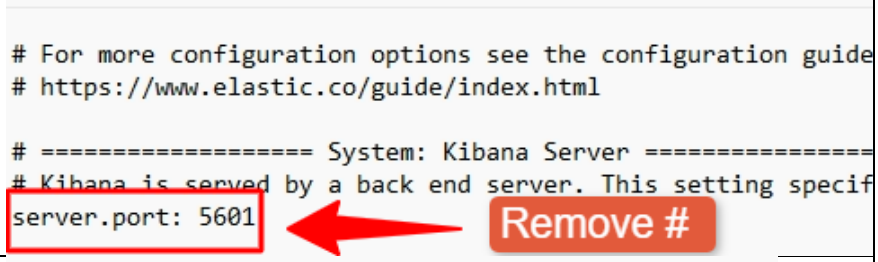
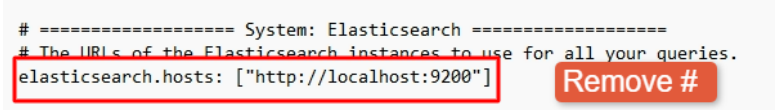
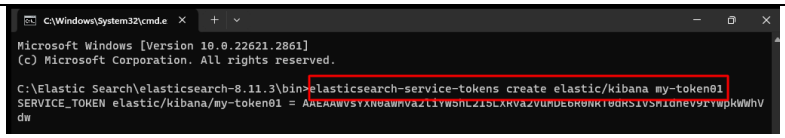
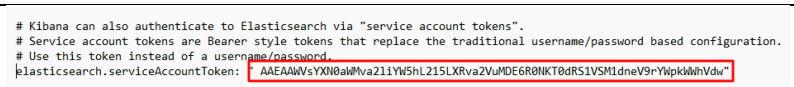
Kibana is an open source browser based visualization tool mainly used to analyse large volume of logs in the form of line graph, bar graph, pie charts , heat maps, region maps, coordinate maps, gauge, goals, timelion etc. The visualization makes it easy to predict or to see the changes in trends of errors or other significant events of the input source.Kibana works in sync with Elasticsearch and Logstash which together forms the so called ELK stack.

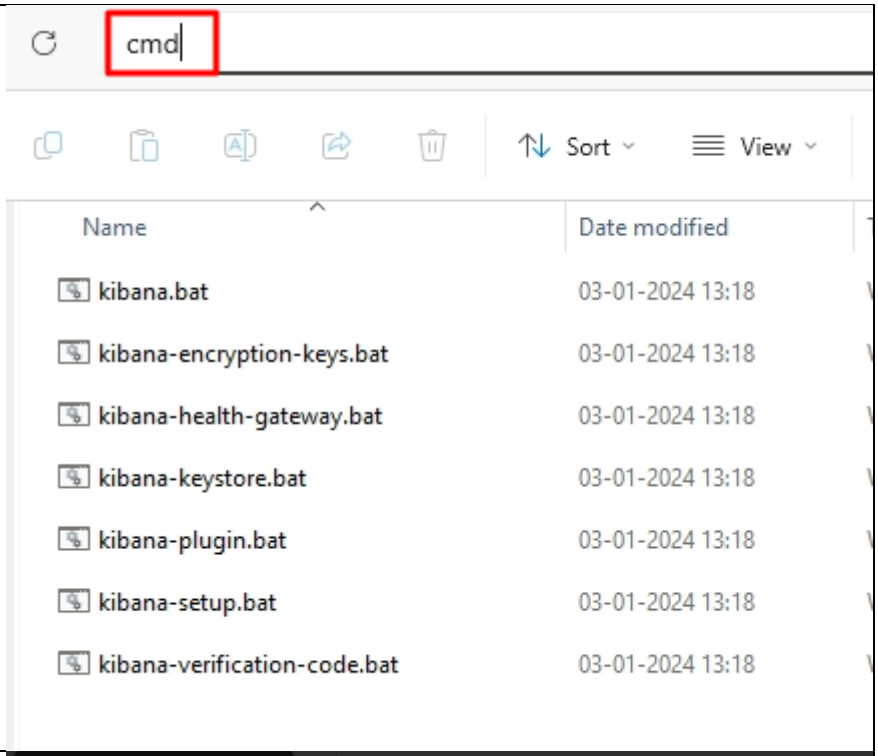
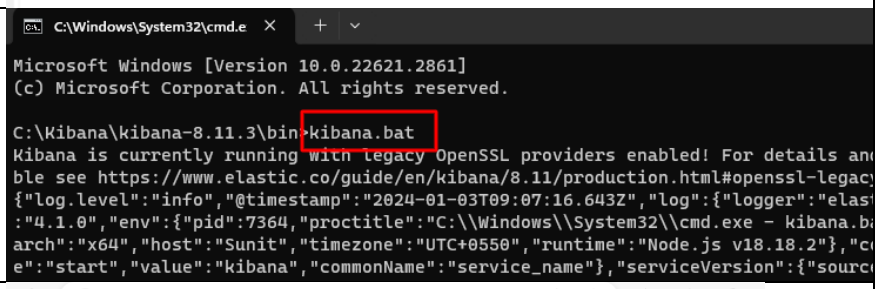
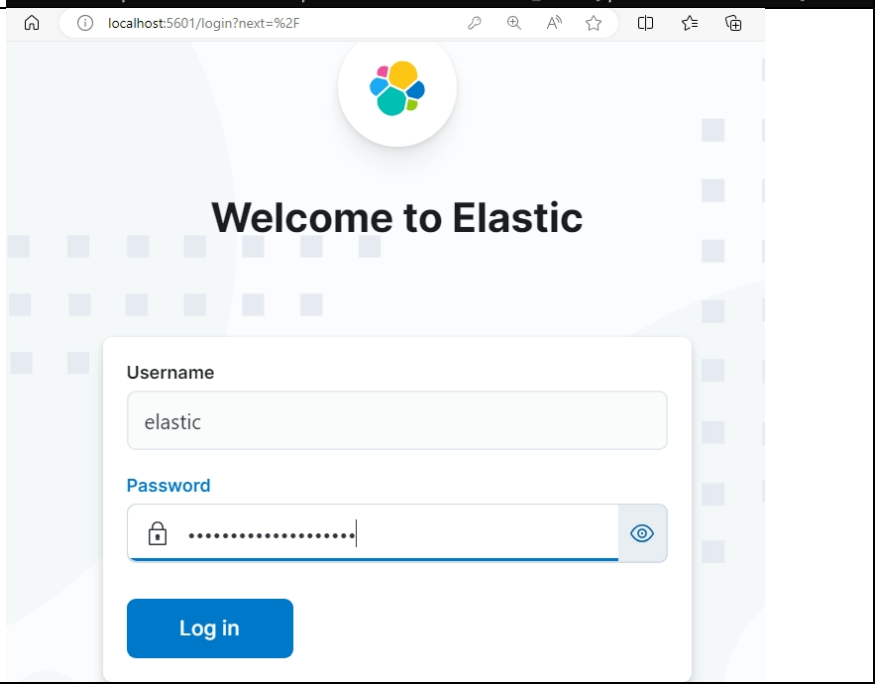
What is Kibana?
Kibana is a visualization tool, which accesses the logs from Elasticsearch and is able to display to the user in the form of line graph, bar graph, pie charts etc.

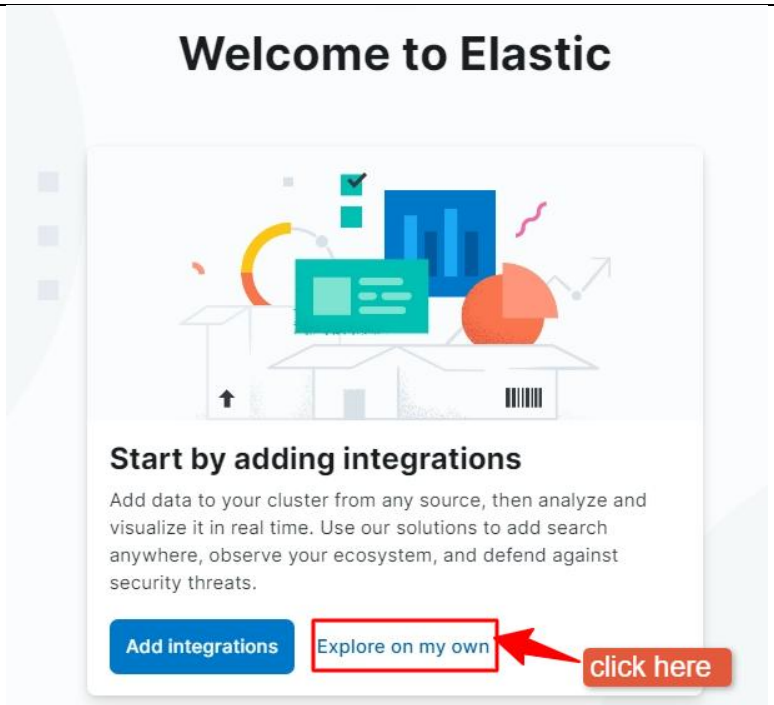
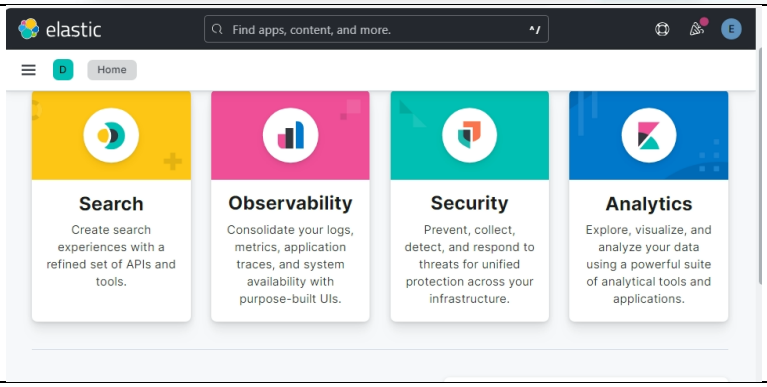


Implementation of Kibana

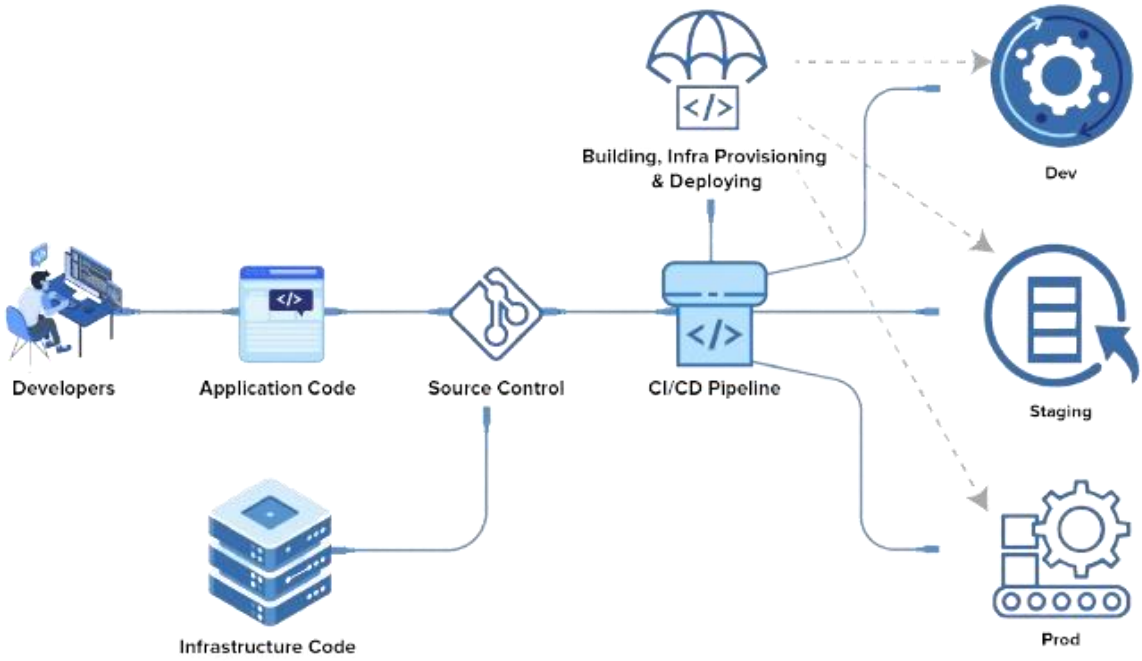
Download **Kibana ZIP** from the following official link: https://www.elastic.co/downloads/kibana	
Unzip the folder in any of the drives.	

Now go inside the Kibana **Config Folder**	
Open **kibana.yml file** in notepad.	
Kibana.yml file inside that enables the server host.	
Now jump to the **Elastic Search Section** in Notepad and enables the **elastic search host**	
Now we have to generate the **Kibana Service Token** So for that come into the elastic search CMD and execute the following command: **Elasticsearch-service-tokens create elastic/kibana my-token01**	
Now copy the token and paste in kibana.yml	

Now go inside the **Kibana bin folder** and launch CMD	
Now in CMD type **kibana.bat** to launch kibana **Note:** before launching kibana your elastic search server should be in **run mode.**	
Now open the browser and type following URL: **Localhost:5601** **Note:** **Username:** elastic **Password:** the password will be the same one you have used at the time of elastic search installation.	

After Login **welcome page** will open.	
Home Page will open.	
For more details, you can watch the video on YouTube.	https://www.youtube.com/watch?v=HckRGumUeDk

Module 5: Automating Deployment with Infrastructure as Code (IaC)



The Infrastructure as Code (IaC) operations have modified how software developers develop, evaluate, and release applications by increasing the number of development and distribution cycles. To make infrastructure development and configuration more competitive and successful, reducing the costs and effort involved, automation tools that facilitate these activities are essential.

Principles of Infrastructure as Code (IAC)

- Easily recreate systems
- Disposable Systems
- Self-Documentation
- Design Is Always Changing
- Generic Modules
- A single, unified API for automated infrastructure deployment
- https://www.xenonstack.com/insights/infrastructure-code-principles

Features of IaC:

- **Automation**: IAC automates the provisioning and configuration of infrastructure, reducing manual errors and saving time.

- **Repeatability**: IAC scripts can be used repeatedly, making it easy to recreate the same infrastructure in multiple environments.
- **Version Control**: IAC code is stored in version control systems like Git, which makes it easy to track changes, revert to previous versions, and collaborate with others.
- **Scalability**: IAC makes it easy to scale infrastructure up or down, adding or removing resources as needed.
- **Transparency**: IAC makes the infrastructure transparent and understandable, as the code defines the infrastructure components and their relationships.
- **Improved Security**: IAC helps ensure that infrastructure is configured consistently and securely, reducing the risk of security vulnerabilities

Applications of IaC:

- Cloud computing
- DevOps
- Continuous integration and delivery (CI/CD)
- Networking
- Web application deployment
- Database deployment
- Big data

Git and Github

Git is a popular version control system. It was created by Linus Torvalds in 2005 and has been maintained by Junio Hamano since then.

It is used for:

- Tracking code changes
- Tracking who made changes
- Coding collaboration

What does Git do?

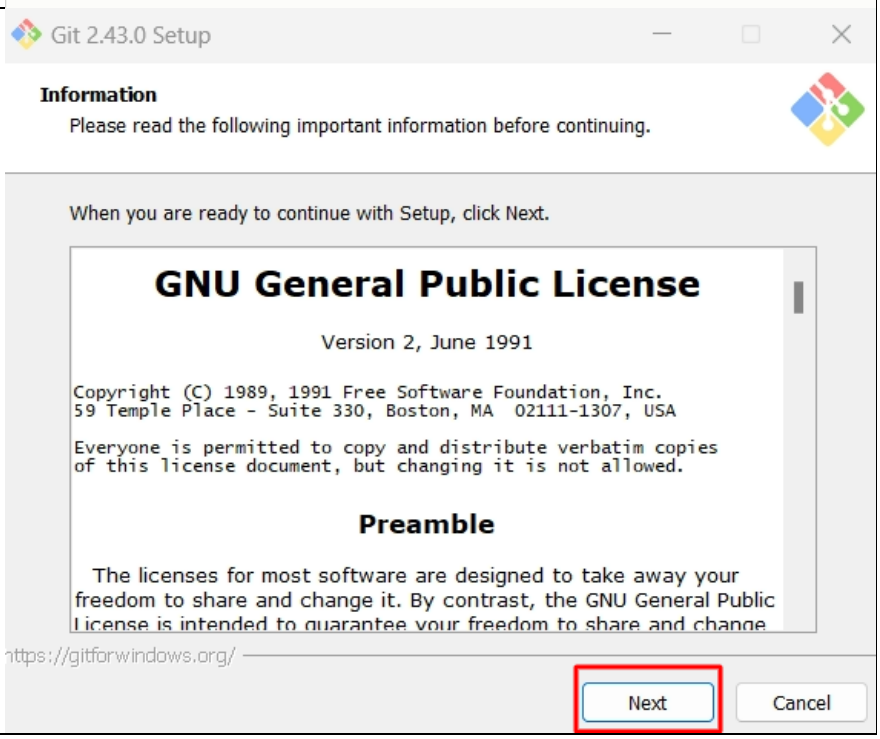
- Manage projects with **Repositories**
- **Clone** a project to work on a local copy
- Control and track changes with **Staging** and **Committing**
- **Branch** and **Merge** to allow for work on different parts and versions of a project
- **Pull** the latest version of the project to a local copy
- **Push** local updates to the main project

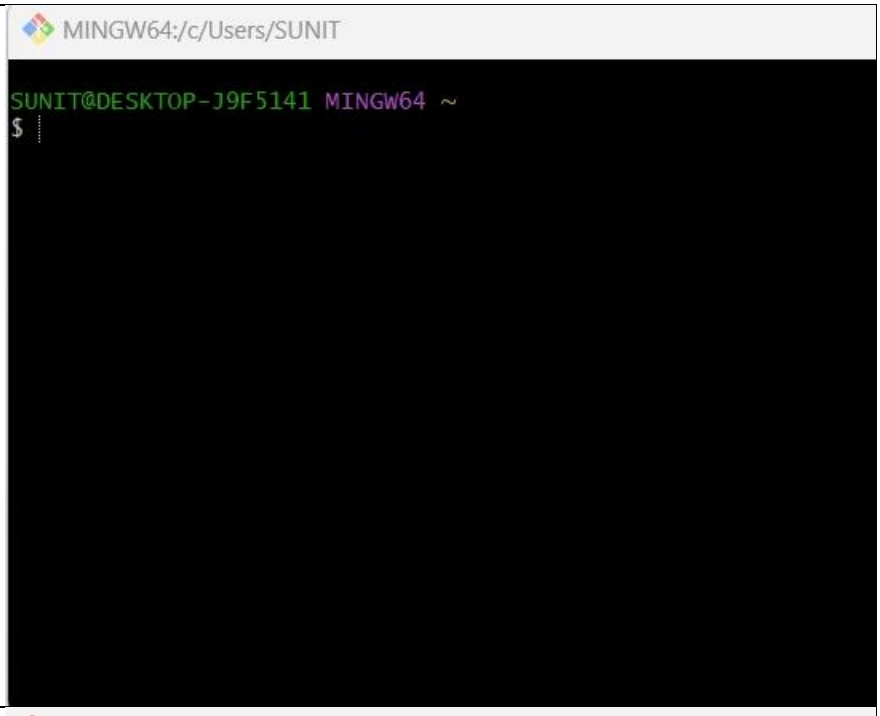
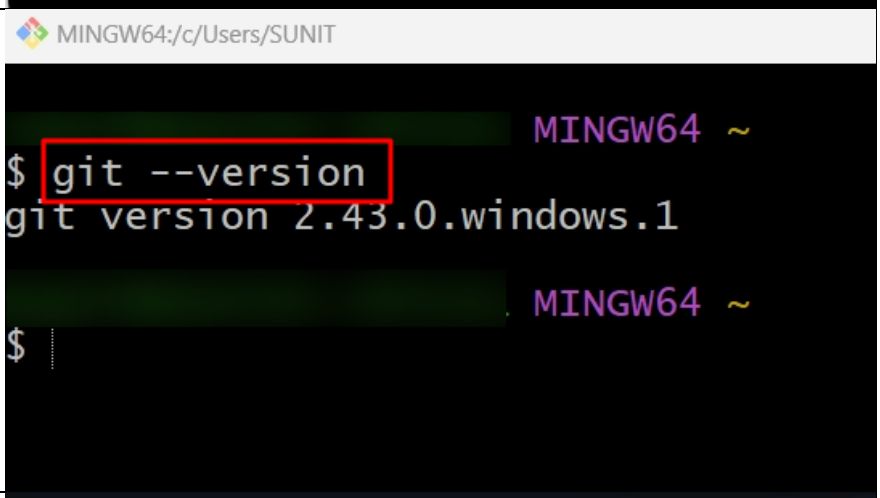
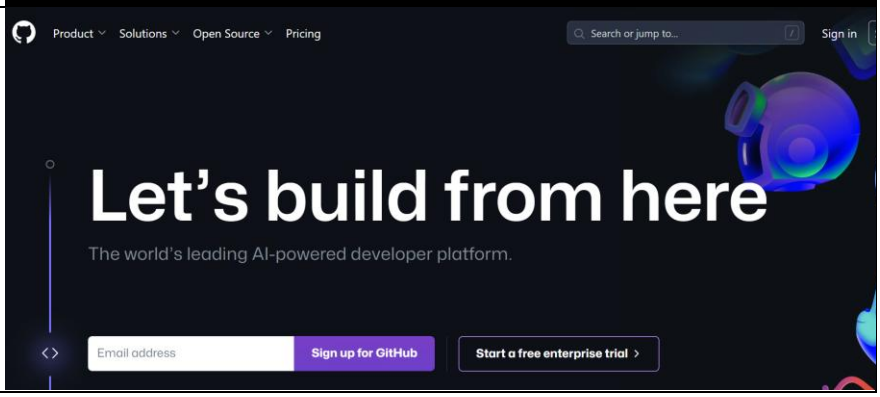
What is GitHub?

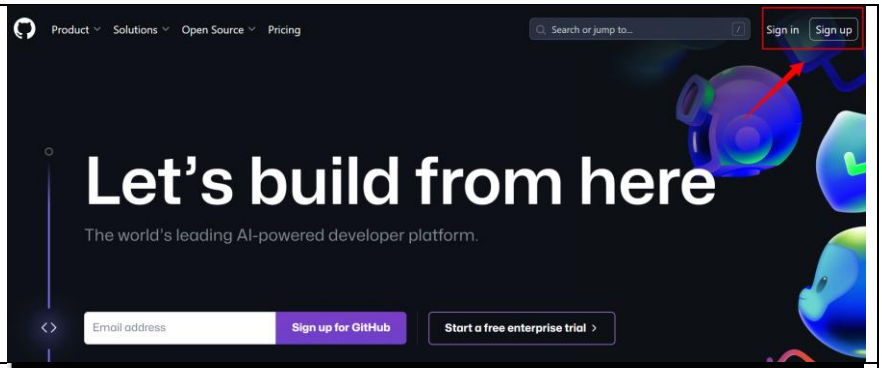
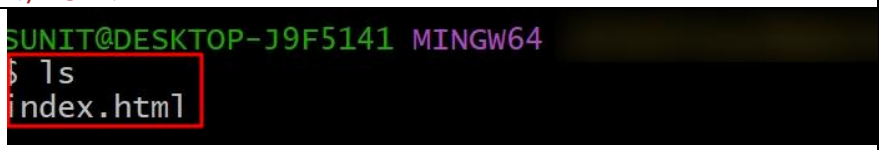
- Git is not the same as GitHub.
- GitHub makes tools that use Git.
- GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018.
- In this tutorial, we will focus on using Git with GitHub.

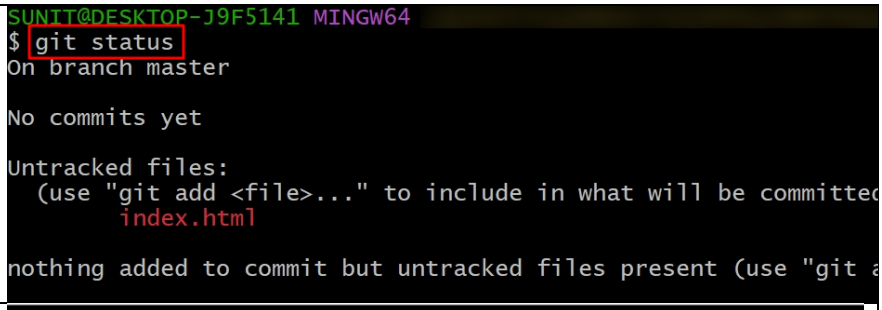
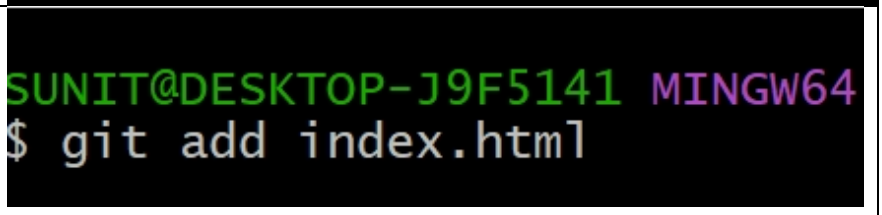
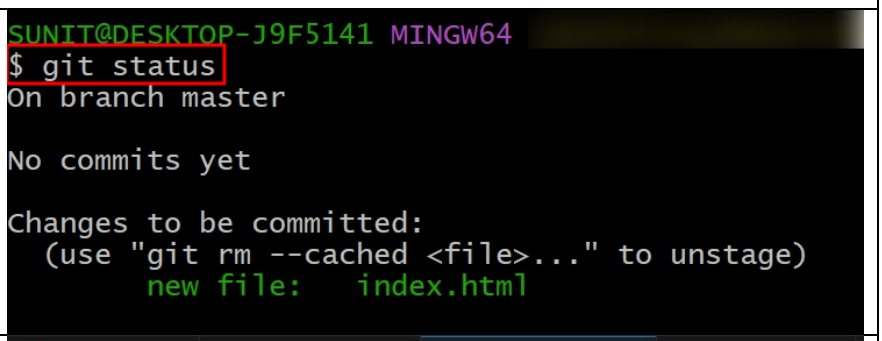
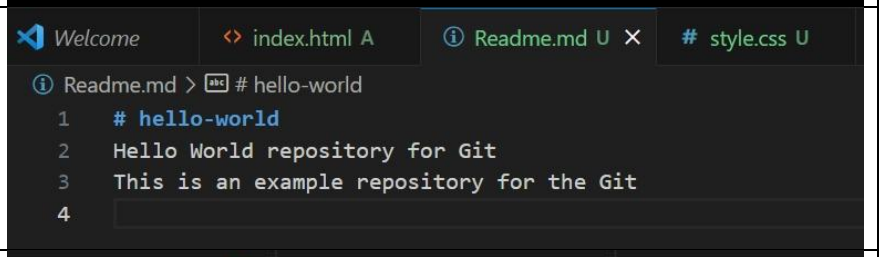
Learning Activity: Implementation of Git and GitHub

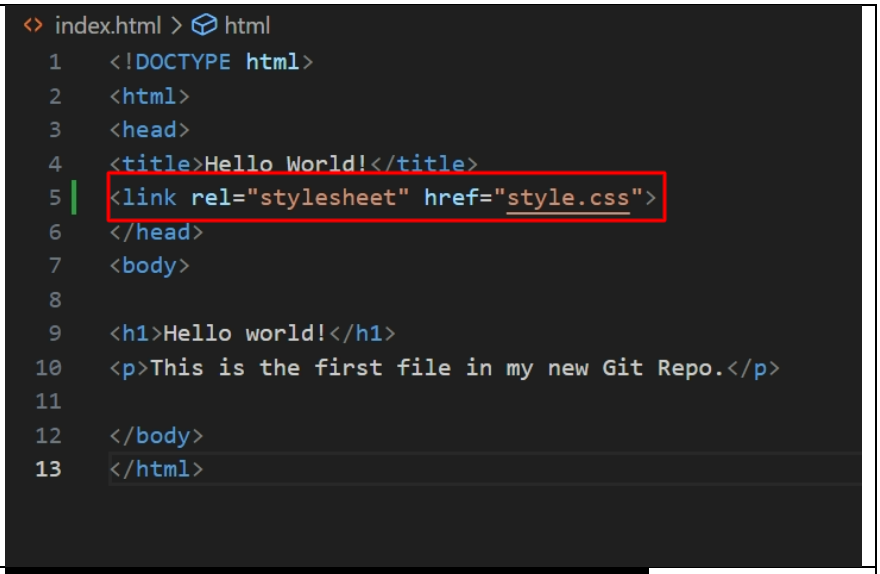
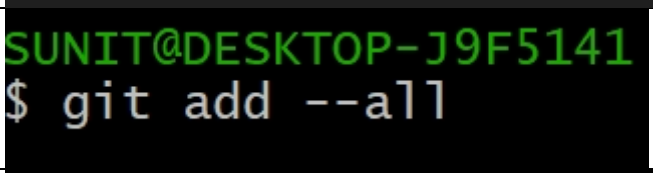
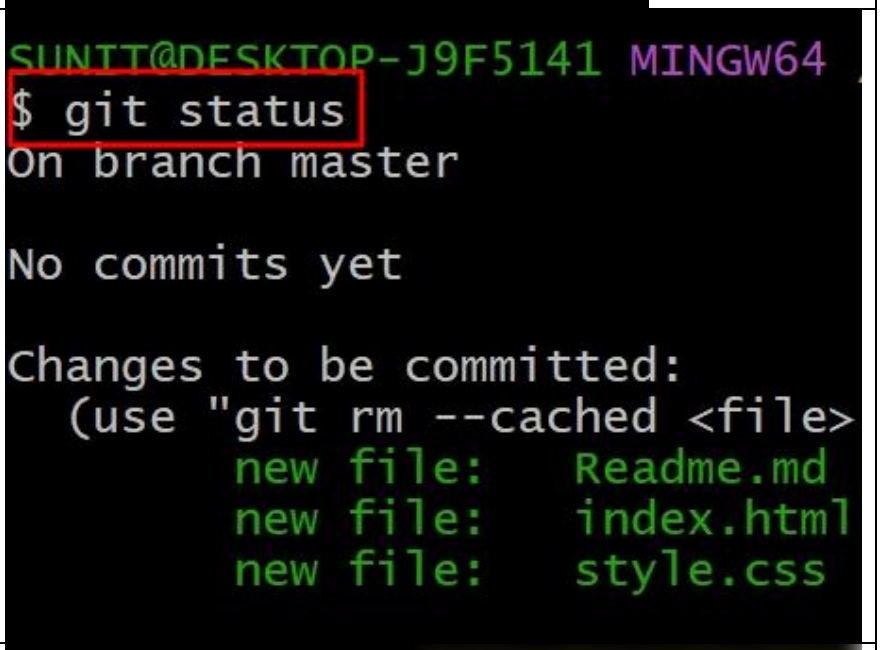
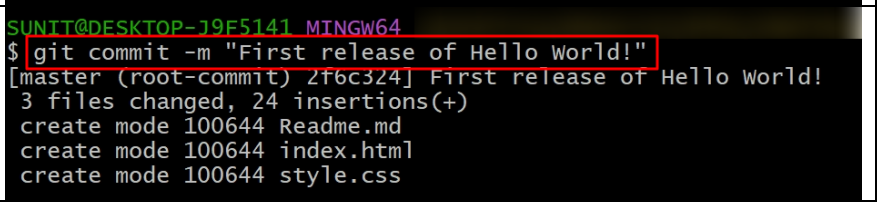
Download the **Git** from the following URL: https://git-scm.com/downloads	
Now Select the OS according to the system. We are working on **Windows** so click on windows	

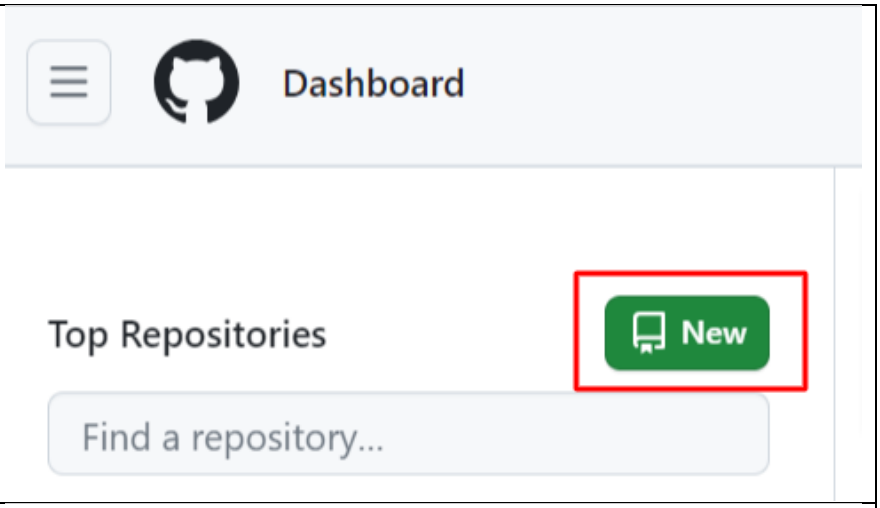
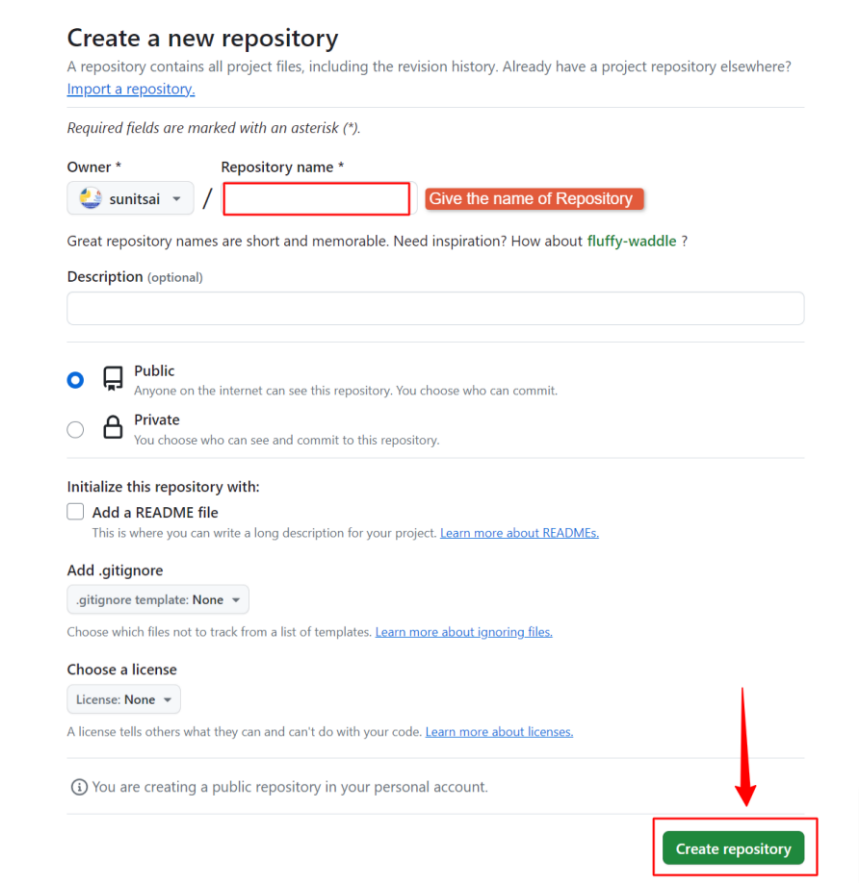
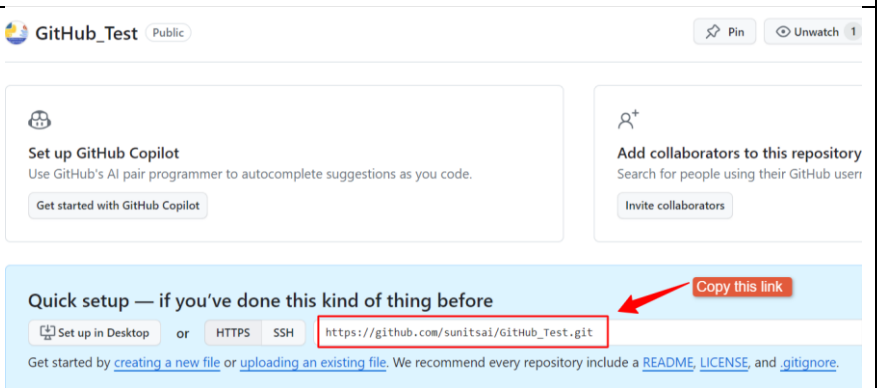
Now select the installer setup	
Now launch the setup of **Git** and follow the wizard just click on the **Next** button and keep the default settings only don't change any configurations.	

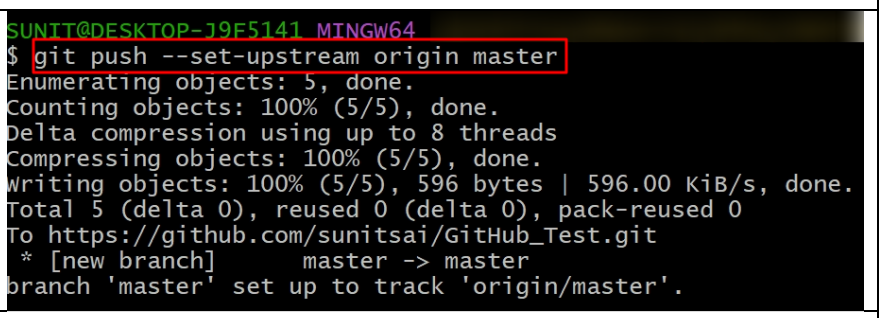
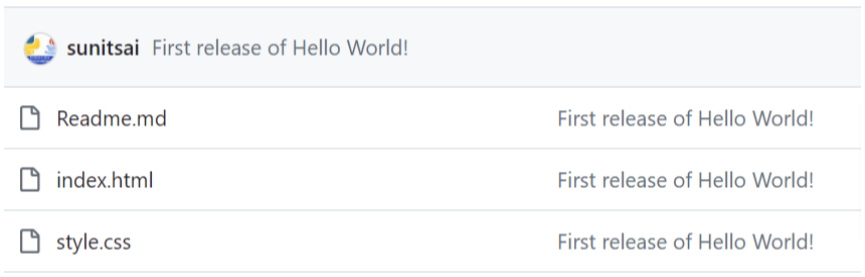
Now open **Git Bash** after the installation is Complete.	
Now check the version of Git type the following command: `git --version`	
Now Open the **GitHub** follow the URL: https://github.com/	

Check the following things: **New User:** Click on **Sign Up** **Existing User:** Click on **Sign In**	
Now configure Git by using the following command: **`git config --global user.name "username"`** **`git config --global user.email "your email"`**	
Now, let's create a new folder for our project	
Now go inside the folder using Git bash	
Once you have navigated to the correct folder, you can initialize Git on that folder	
For this example, I am going to use a simple HTML file like this	`<!DOCTYPE html>` `<html>` `<head>` `<title>Hello World!</title>` `</head>` `<body>` `<h1>Hello world!</h1>` `<p>This is the first file in my new Git Repo.</p>` `</body>` `</html>`
Let's go back to the terminal and list the files in our current working directory	

Then we check the Git **Status** and see if it is a part of our repo	
we are done working with **index.html**. So we can add it to the Staging Environment	
The file should be **Staged**. Let's check the status.	
A **Readme.md** file that describes the repository	
Now add CSS file in the folder.	

Update the CSS link in index.html file	
Now add all files in the current directory to the Staging Environment	
Then we check the Git **Status** and see if it is a part of our repo	
Now **commit** the file into a repo.	

Now open GitHub and click on New button to create a repository.	
Give the name of repository and click on create.	
Now copy the repository link.	

Open the Git Bash and let add the three file in our repository. To add the files in repository follow the figure.	
Now to push the file into the repo.	
Now refresh the GitHub page and files will be uploaded there with the first commit.	

Module 6: Security and Compliance in CI/CD Pipelines

CI/CD Pipeline:

- A continuous integration and continuous delivery/deployment (CI/CD) pipeline is a series of steps that software delivery undergoes from code creation to deployment.
- CI/CD encompasses a series of automated processes — from code development to production deployment — that enable frequent and reliable delivery of code changes to the production environment. It forms the backbone of DevOps, a shift in software development that emphasizes collaboration between development and operations teams to ultimately shorten the development lifecycle without compromising software quality.

Integrating security practices into CI/CD pipelines:

- it's important to include security checks at various stages to ensure that your code is secure and compliant with security standards. There are a number of measures you can take to secure your CI/CD pipeline.

Planning phase:

- This stage involves gathering requirements and consumer input to develop a product roadmap. It also encompasses the best practices and policies for a successful DevOps strategy.
- You should also take advantage of threat modeling to help identify potential areas of attack and take steps to secure your pipeline. In threat modeling, security vulnerabilities are identified, and countermeasures are determined to mitigate them. By applying threat modeling to CI/CD pipelines, you can identify potential attack areas and take measures to secure them.
- Supply-chain Levels for Software Artifacts (SLSA) is also useful during the planning phase. This security framework comprises a checklist of standards and controls to prevent supply-chain attacks, safeguard against integrity challenges, and safeguard software packages and infrastructure in your organization.

Coding phase:

- In the coding phase, the developers write the necessary code to build the software. The code must be written by predefined standards and design guidelines.
- You should use source code scanners such as CAST Application Intelligence Platform (AIP) or CodeSecure to detect pieces of code that might be vulnerable to security threats.

Build phase:

- During the build phase, the developers are responsible for committing their source code to a shared repository. Once the code changes are checked into the repository, builds are triggered, and automated tests are executed to verify if the builds comply with the requirements.
- Here are some tips for setting up security checks in your CI/CD pipeline:

- Include a static code analysis tool in your build stage to check the code for common security vulnerabilities and compliance issues.
- Use Static application security testing (SAST) tools like SonarQube, Veracode, AppScan, or Codacy.
- Use software composition analysis (SCA) tools such as Veracode, Sonatype Nexus Platform, etc.
- Set up security-related test cases based on organization policies. These tests can check for things like cross-site scripting (XSS) and SQL injection flaws.
- Use a code-signing service to sign your code ahead of deploying the code to the production environment. This will help ensure that the code has not been tampered with and that it comes from a trusted source.

Testing phase:

- Once a build is successful, the software is tested to detect any potential bugs. If new features are added, a new build is generated and regression testing is performed on the new build to verify if the functional tests succeed.
- At this stage, you should run container scanning tools (e.g., Datadog, Clair, Anchore, and Qualys) or dynamic analysis security testing (DAST) tools (e.g., Netsparker and Acunetix)

Deployment phase:

- In this phase, the build is deployed to the production environment.

Monitoring Phase:

- During this phase, the build is monitored to ensure that it works as expected. The application deployed in the production environment is observed to evaluate performance and other aspects.

What is Vulnerability Scanning?

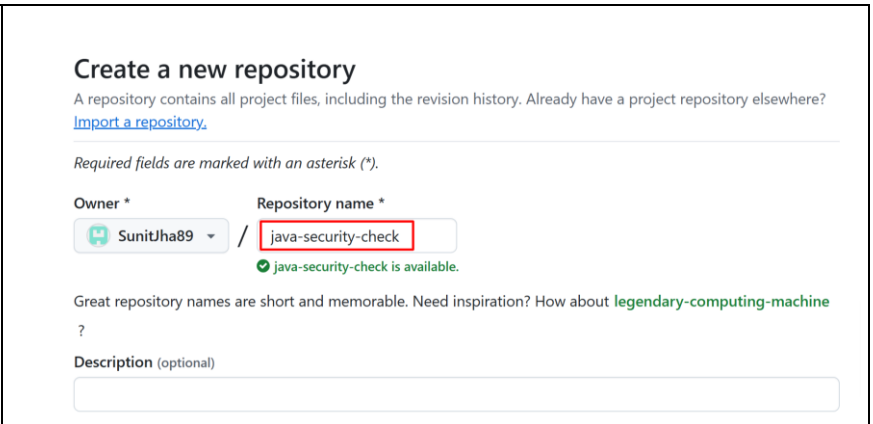
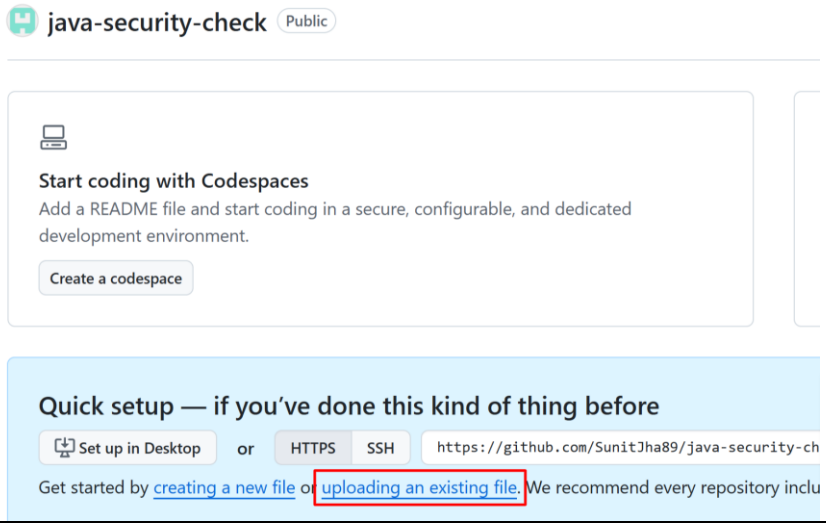
- Vulnerability scanning is performed to detect and remediate these vulnerabilities.
- Vulnerability scanning can be done either by the team or by automated software to manage different types of vulnerabilities. Automated vulnerability scanning is different from manual vulnerability scanning, in which a human examines an application or system and searches for vulnerabilities.

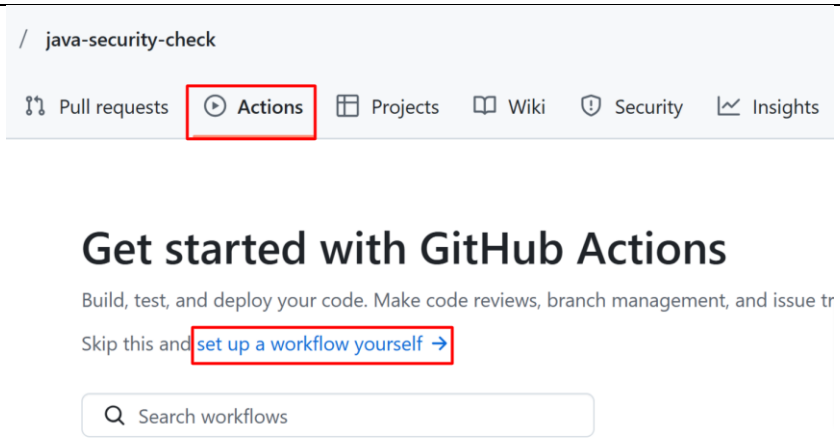
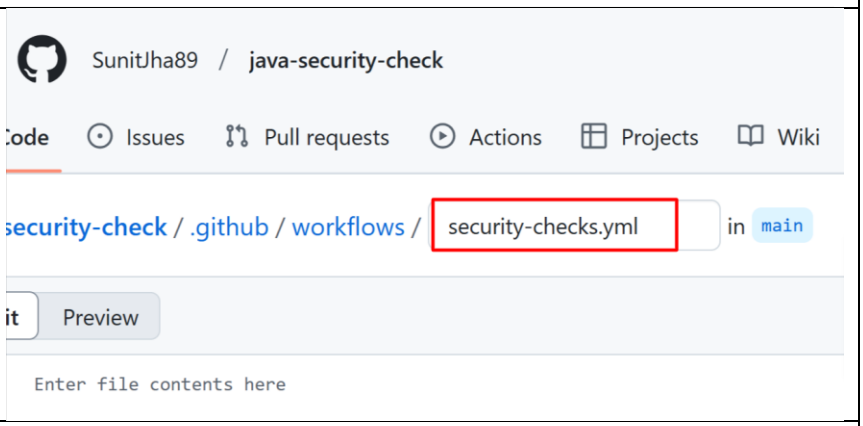
CI/CD Secrets Management:

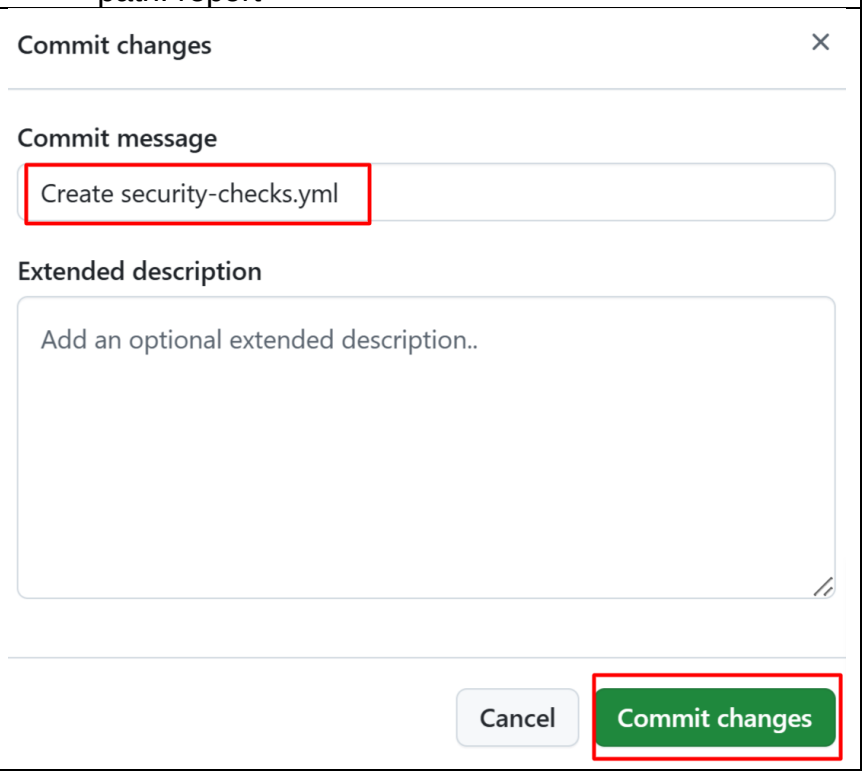
- A continuous workflow calls for unique practices when it comes to managing vault secrets and privileged access. Secrets refer to user credentials, keys, passwords, private certificates, and anything else that is essential for software deployment and must be kept secret.
- Secrets can serve as authentication tools to prove the identity of a user or system or authorization, where access to certain corporate resources is granted based on user privileges.
- When working with CI/CD, organizations must use multiple services, environment repositories, cloud providers, and verification providers. Keeping accurate secrets for each team and each service can be a challenge when everyone needs a different level of access.

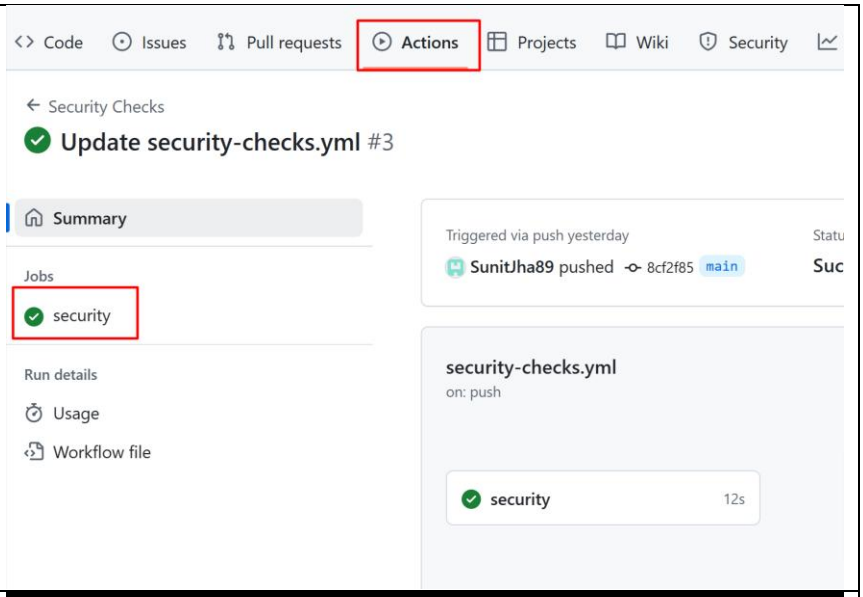
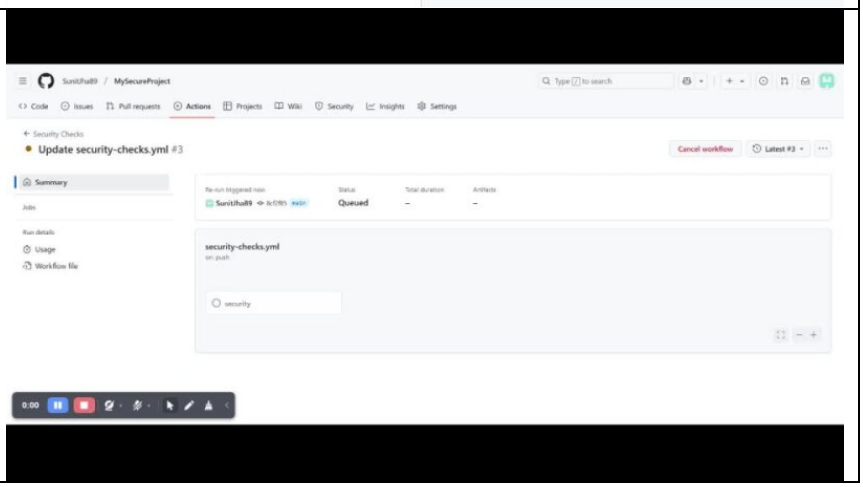
- Also, part of development is authenticating digital tools to get the most out of them. Any resources you use must have the right access to do their jobs, especially in hybrid cloud environments or automated tools like Kubernetes.
- All this complexity makes secrets management in a CI/CD environment expensive and error-prone if managed improperly. Because secrets are required for any CI/CD working environment, it's important to pick up some best practices.

Learning Activity: Implementing automated security checks and vulnerability scans.

1. Create a repository name "Java-Security-Checks".	
2. Create a "HelloWorld.java" file.	public class HelloWorld { public static void main(String[] args) { System.out.println("Hello, World!"); } }
3. Upload the java file in GitHub repository.	

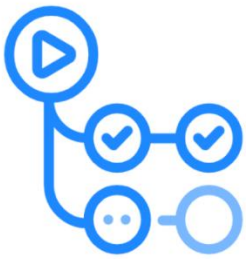
4. Click on **Action** and then click on “Set Up a Workflow Yourself” for creating a yml file.	
5. Give the name of YML file.	
6. Write a Workflow in file name “security-checks.yml”	name: Security Checks on: push: branches: - main pull_request: branches: - main jobs: security: runs-on: ubuntu-latest steps: - name: Checkout code uses: actions/checkout@v3 - name: Set up JDK 11 uses: actions/setup-java@v3 with: distribution: 'adopt' java-version: '11' - name: Run Security Scan

	run: | curl -LO https://github.com/jeremylong/DependencyCheck/releases/download/v6.5.3/dependency-check-6.5.3-release.zip unzip dependency-check-6.5.3-release.zip ./dependency-check/bin/dependency-check.sh --project JavaSecurityCheck --scan ./ --format HTML --out report --disableAssembly -v - name: Upload Report uses: actions/upload-artifact@v3 with: name: security-report path: report
7. Commit the workflow file in main branch.	

8. Check the workflow by clicking on action button.	
9. Now check the security check done by github action.	

Module 7: Hands-on Workshop: Building and Managing CI/CD Pipelines

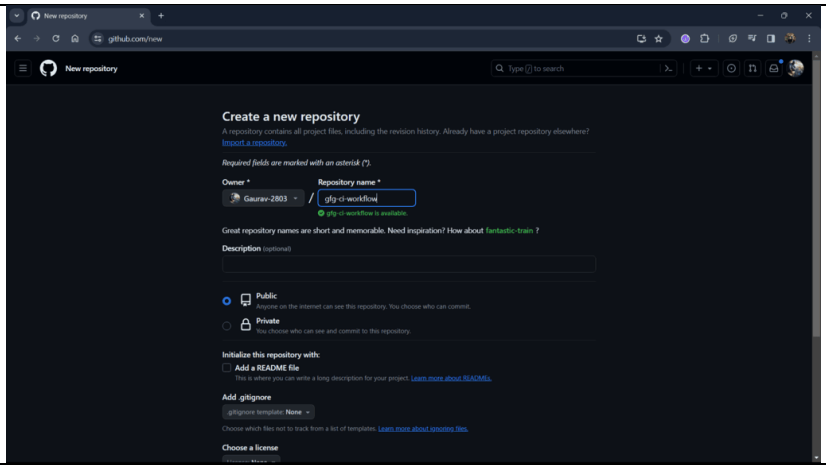
Collaborative exercises: designing, building, and managing CI/CD pipelines.



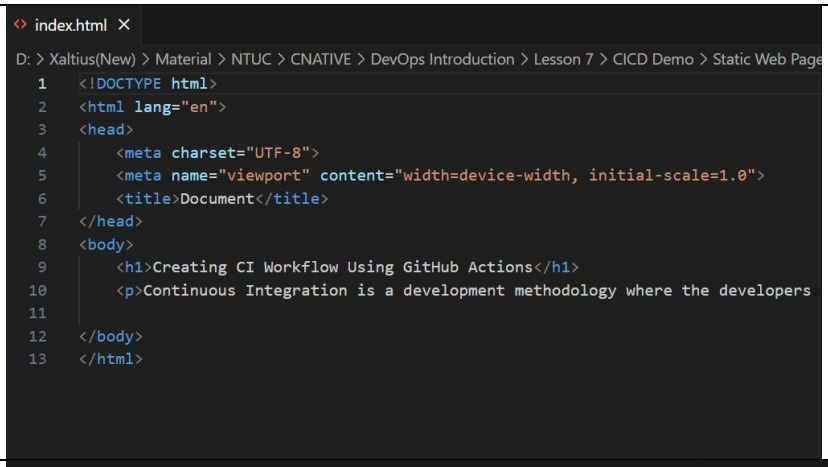
GitHub Actions

GitHub Actions is a continuous integration and continuous delivery (CI/CD) feature provided by GitHub that allows you to automate your build, test, and deployment pipeline whenever any changes happen in your repo.

Learning Activity: Creating CI Workflow Using GitHub Actions: A Step-By-Step Guide

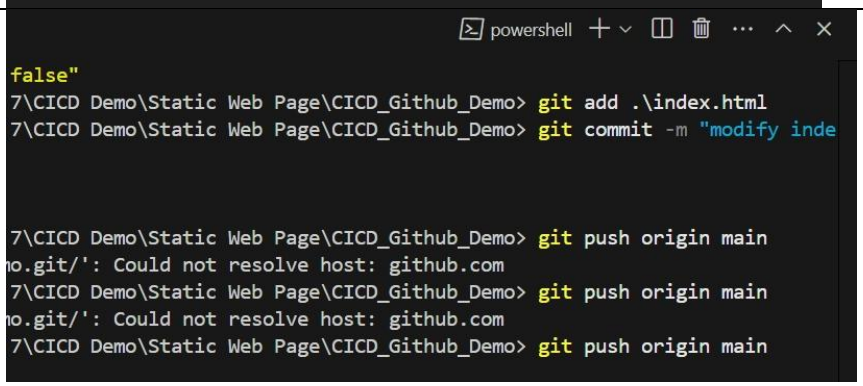
1. Create an Empty **GitHub Repository.**	

2. Open your project in any text editor. Here I'm taking basic **static HTML** file with 1 heading and 1 paragraph. You can take anything.

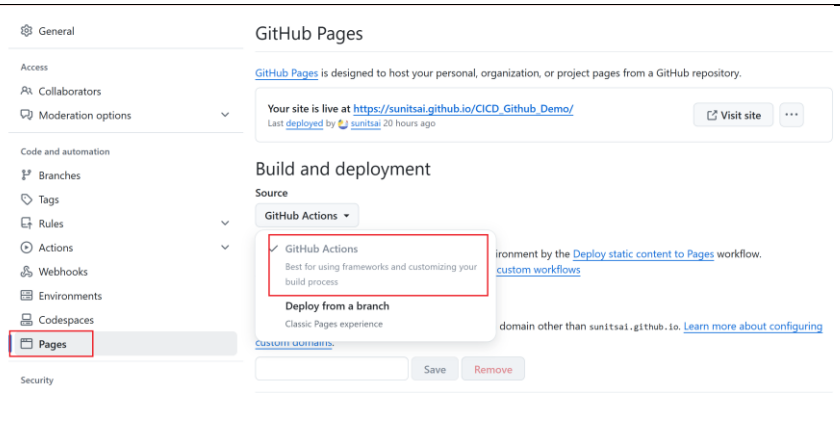


3. Open Git terminal and **Push** your project on the GitHub Repository using following steps.

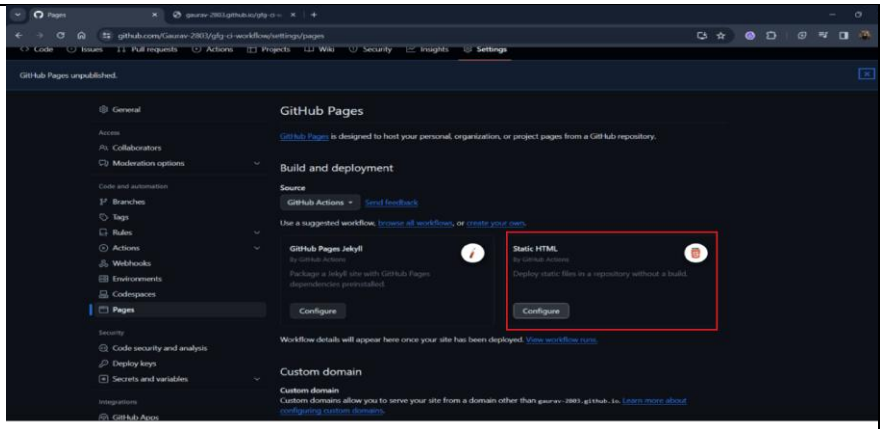
- Initalize empty Git repo using **git init**
- Add files to staging area using **git add .**
- Commit your changes using **git commit -m "your message"**
- Create remote origin using **git remote add origin "repo link"**
- Push the project using **git push -u origin main**



4. Open the **settings** of your repository and go to **Pages**. From the source select **GitHub Actions.**



5. Configure a **Static HTML**. This will generate a pre-build workflow for your static HTML file. GitHub provides in-built workflow, for my purposes I'll select static HTML you can choose yours respective workflow.

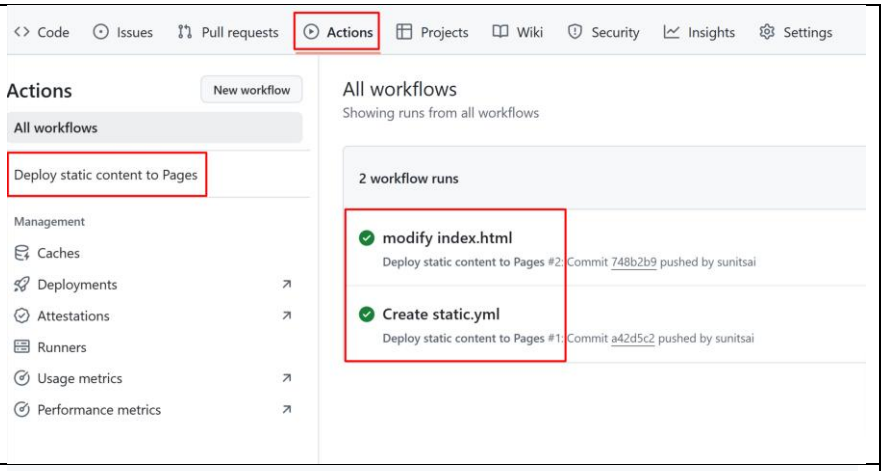
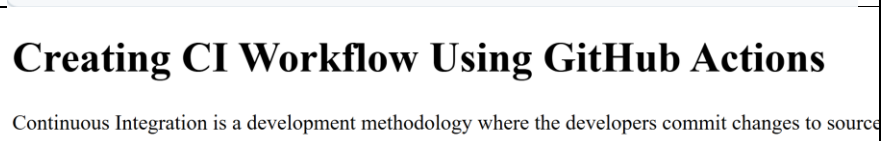


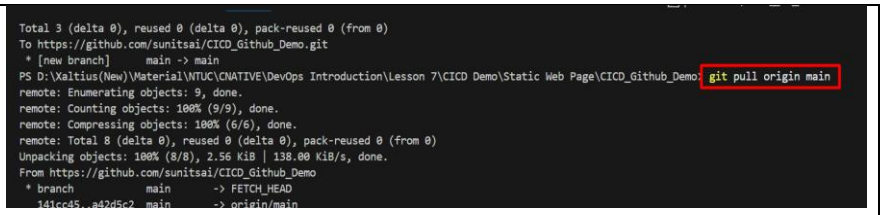
6. The following image is the workflow **YAML file** that will automate your process of deployment.

main CICD_Github_Demo / .github / workflows / static.yml

Blame 43 lines (38 loc) · 1.22 KB Code 55% faster with GitHub Copilot

```yaml
  # Runs on pushes targeting the default branch
  push:
    branches: ["main"]

  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:

# Sets permissions of the GITHUB_TOKEN to allow deployment to GitHub Pages
permissions:
  contents: read
  pages: write
  id-token: write

# Allow only one concurrent deployment, skipping runs queued between the run in-progress and latest queued.
# However, do NOT cancel in-progress runs as we want to allow these production deployments to complete.
concurrency:
  group: "pages"
  cancel-in-progress: false

jobs:
  # Single deploy job since we're just deploying
  deploy:
    environment:
      name: github-pages
      url: ${{ steps.deployment.outputs.page_url }}
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Setup Pages
        uses: actions/configure-pages@v5
      - name: Upload artifact
        uses: actions/upload-pages-artifact@v3
        with:
          # Upload entire repository
          path: '.'
      - name: Deploy to GitHub Pages
        id: deployment
        uses: actions/deploy-pages@v4
```

7. From the above repo tabs go to **Actions** and check if your app is deployed on **Github Pages** or not.	
8. Go to the **link** provided by GitHub this is where your HTML file is deployed.	
9. After opening the link you can see the contents of your HTML file.	

10. Now let's check if CI workflow or not. First, pull your changes. • Modify your code and push it on the GitHub. • For every push our workflow will be generated and changes will reflect on your page.	
11. **Wait** for 10-15 seconds and **reload** your site. The new changes will appear on the site.	**Creating CI Workflow Using GitHub Actions** Continuous Integration is a development methodology where the developers commit changes to source code **What Are Github Actions?** GitHub Actions is a platform built into GitHub that automates all the SDLC steps like development, testing,

Applying best practices for testing, deployment, and monitoring

- **Prioritize security, testing, and time to release:**
  Before configuring your CI/CD pipeline, prioritize security, testing, and time to release. This determines how you approach each step in your pipeline. For example, if security is a top priority, you can implement more controls on your code before it's deployed.

- **Test code in the early stages:**
  You can catch bugs and identify issues at different testing stages. For example, if developers find a problem on their local machine, they can fix issues in the staging environment before moving the code to production.

- **Identify tests that can be automated:**
  Any test that doesn't require human interaction should be automated and integrated into the pipeline as a rule of thumb. You should adopt an "automate everything" mindset but also understand not all tests can be automated.

- **Commit daily:**
  High-performing teams can commit to their repository once a day. This requires extensive automation to ensure high quality with each new version, but it's well worth the effort.

- **Choose tools that support your priorities:**
  When choosing DevOps tools, consider what matters most to you as a developer and business analyst. Do you heavily focus on infrastructure automation? Does an application lifecycle management (ALM) methodology drive your work? Do you manage multiple virtual machines? Consider vendor options designed explicitly for CI/CD operations to address these fundamental concerns.

- **See if you're ready to adopt a microservices architecture:**
  Microservices architecture is a way to structure software applications so that each component functions, updates, and scales independently.

- **Use on-demand testing environments:**
  The ease of creating a new testing environment at the click of a button helps run tests as early and often as possible. As a result, developers can make the most of their time knowing they'll complete their work quickly.

- **Involve the whole team in the CI/CD implementation:**
  Involving each team member in the CI/CD process is critical to CI/CD implementation. CI/CD isn't technical; it's cultural. Make it clear that CI/CD isn't just a technical step but an essential part of the company culture to seek active participation in the process.

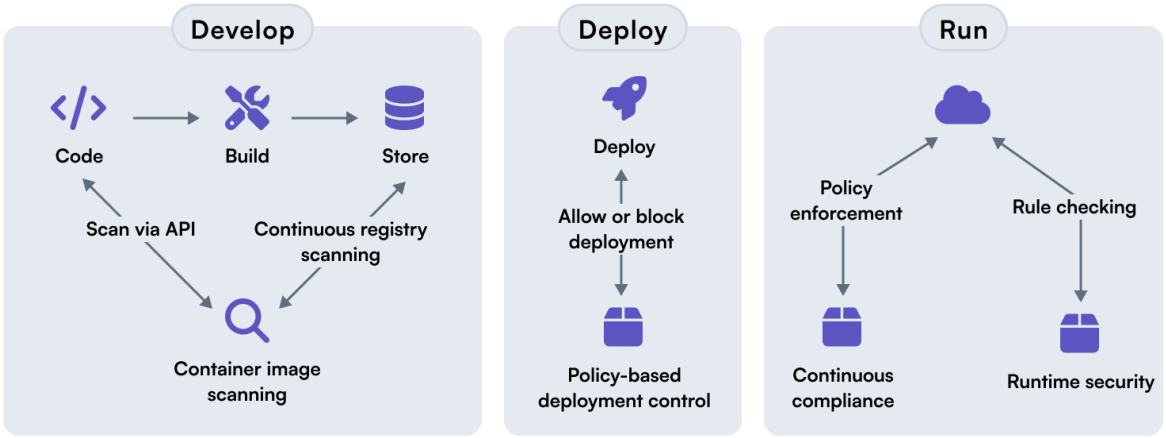
Integrating security and compliance checks into the pipeline.

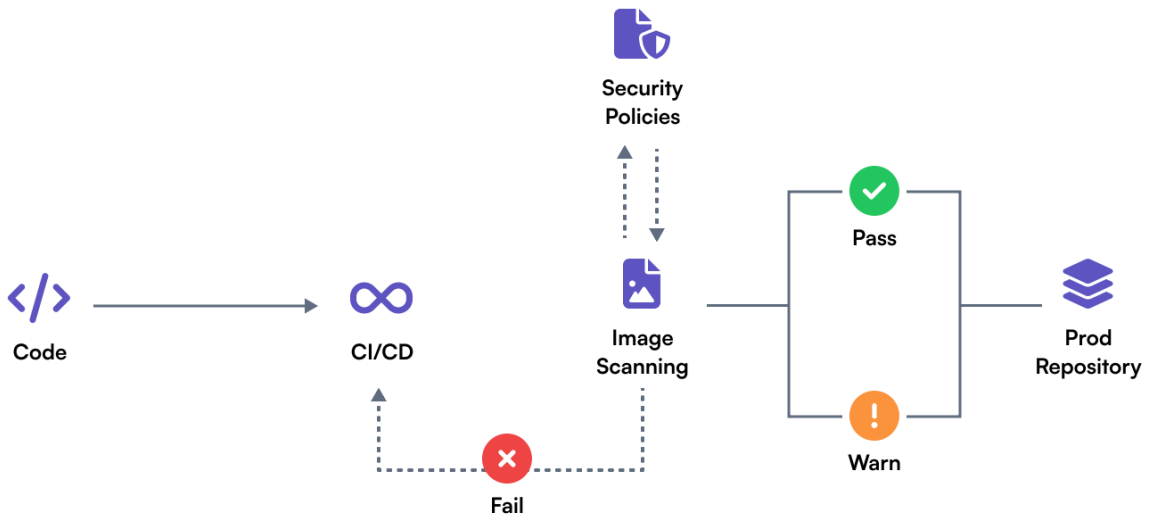
Choosing the right security tools is pivotal for effective integration into the CI/CD pipeline. The selection should be based on the technology stack, development environment, and specific security requirements of the project.

Consider the following when selecting security tools:

- Compatibility: Ensure the tools integrate seamlessly with the existing CI/CD infrastructure.
- Comprehensiveness: Select tools that cover a wide range of security aspects, from code analysis to runtime protection.
- Usability: Tools should be user-friendly, providing clear insights and actionable recommendations.
- Performance: Evaluate the impact on build and deployment times to maintain efficiency

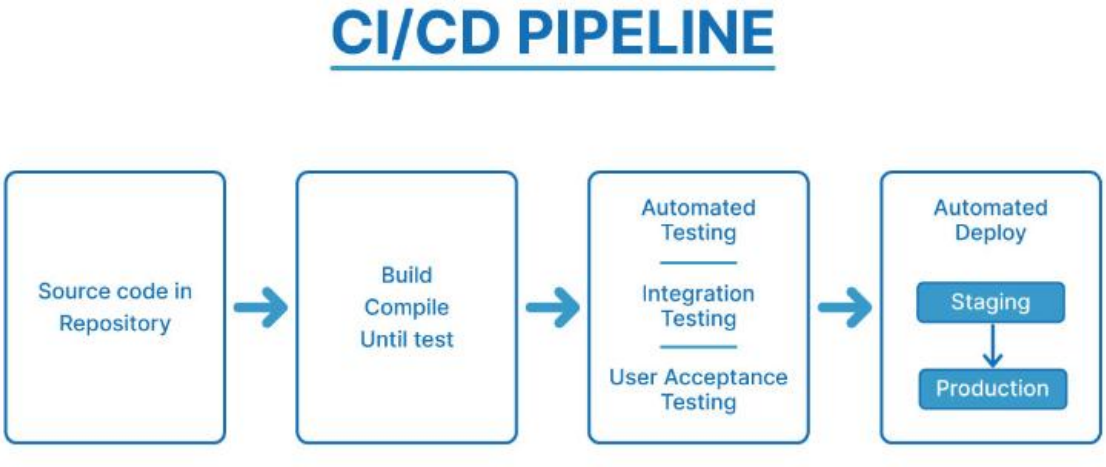
Container Security





Module 8: Use Cases: Successful CI/CD and Continuous Monitoring Implementations

Analyzing real-world use cases of successful CI/CD and monitoring implementations:



CI/CD pipelines real-life example:

There are many big giant companies nowadays implementing DevOps. Netflix and Facebook are the prime examples that have successfully implemented CI/CD pipelines to deliver software updates and features quickly and with high quality. Netflix uses a custom-built platform called Spinnaker to manage its CI/CD pipelines. Spinnaker allows Netflix to deploy code changes to production environments in minutes rather than hours or days.

Case Study 1: Accelerating Software Delivery with Continuous Integration:

- **Background**:
  The case study begins with a well-established e-commerce company facing a common challenge in the digital era: the need to innovate and deliver new features and updates to its online store rapidly. In a highly competitive market, speed to market and agility were critical factors that could set them apart from their competitors. However, their existing software development process was plagued by slow manual testing and deployment procedures, leading to delays in delivering new features and bug fixes to customers.

Identifying scenarios where CI/CD and monitoring practices excel.

Scenario: Organizing a School Science Fair:

The class is creating an Excel sheet to organize the science fair participants, their projects, and scores from judges. Everyone works together to keep the sheet updated and error-free. CI/CD helps them automate some of the work.

Learning Activity: CI/CD Pipeline using Excel:

1. Create one Excel File name "ScienceFair.xlsx"

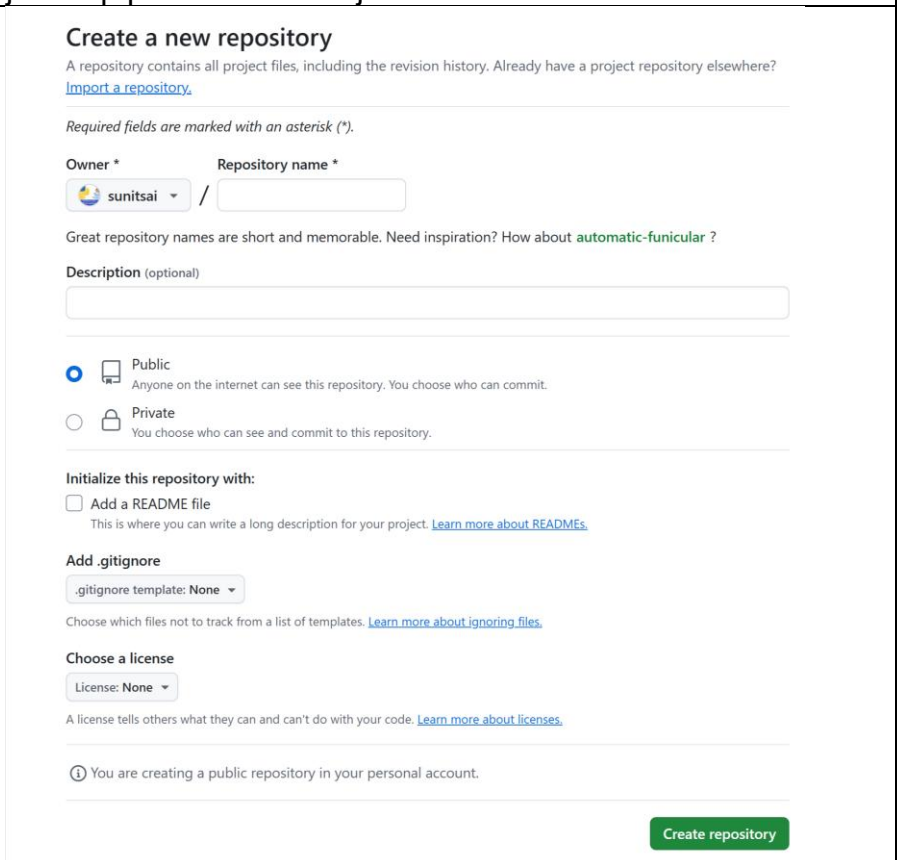
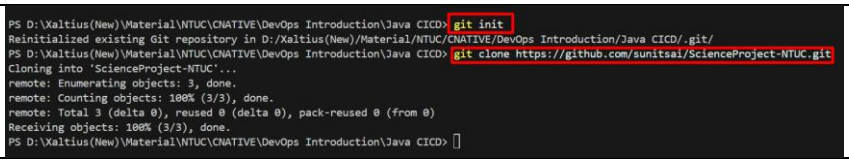
Student Name	Project Title	Judge 1 Score	Judge 2 Score	Average Score
Alex Smith	Solar Power Car	8	9	8.5
Emma Johnson	Volcano Experiment	10	7	8.5
Liam Brown	Water Purification	6	5	5.5
Olivia Garcia	Growing Plants Faster	9	8	8.5
Ava Martinez	Balloon Rocket Race	10	10	10
	Simple Machine Project	7	9	
Sunit		9	8	

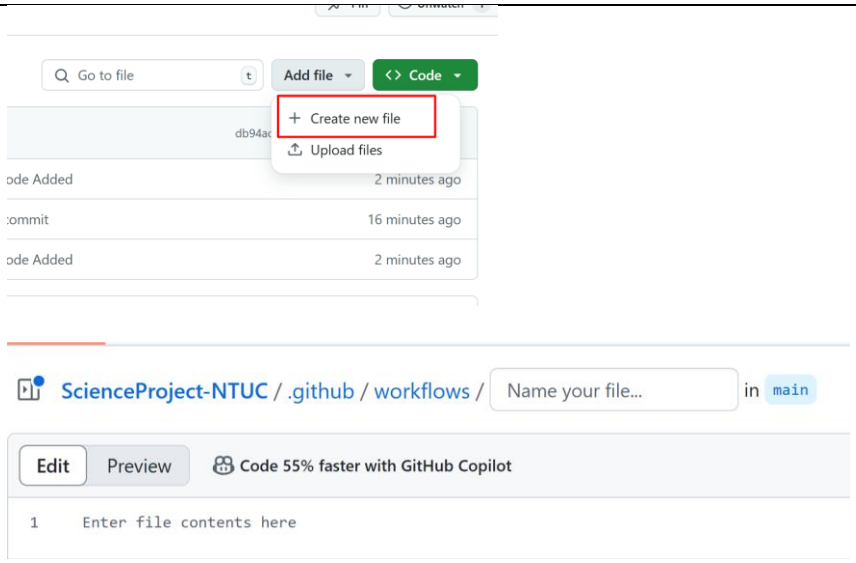
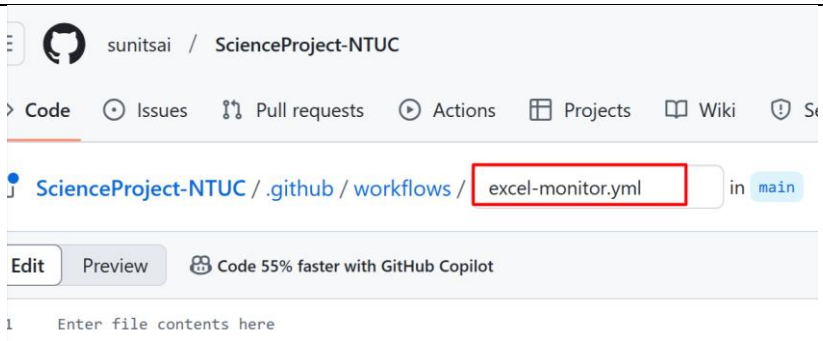
2. Create a Java Maven Project and add dependency in "pom.xml"

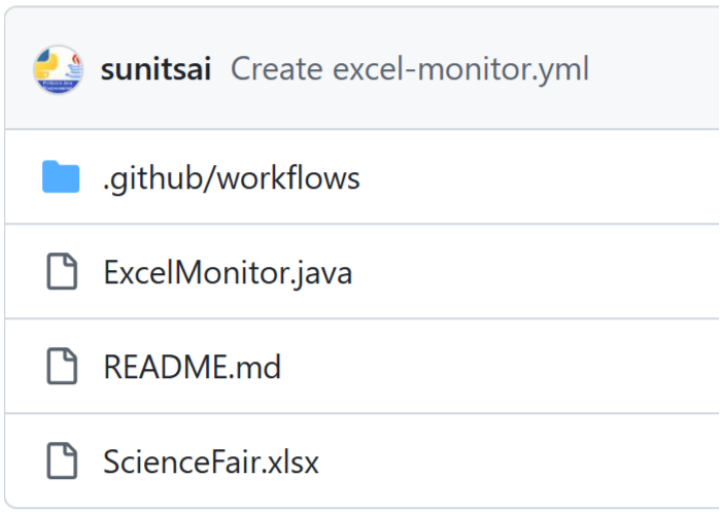
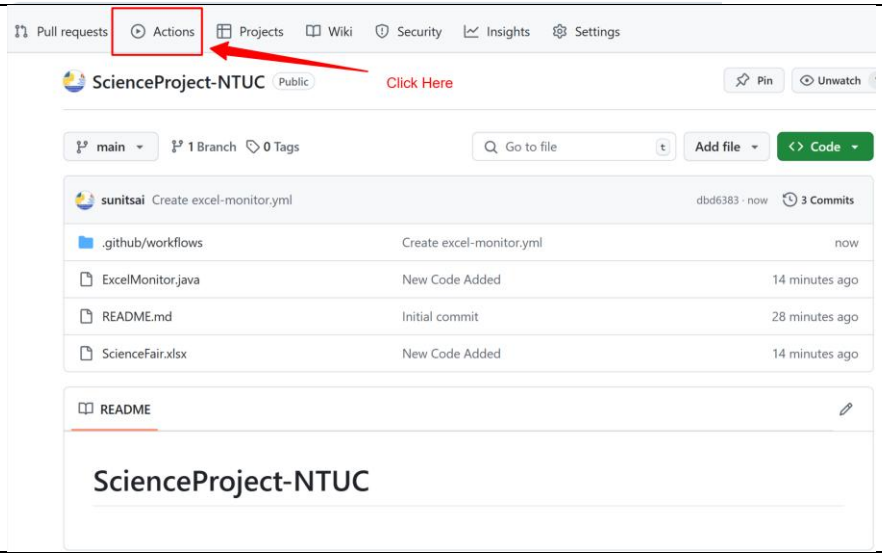
```
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>5.2.3</version> <!-- Check for the latest version -->
</dependency>
```

3. Create a “ExcelMonitor.java” file and write the following code.	package ScienceFair.ScienceFair; import java.io.FileInputStream; import java.io.IOException; import org.apache.poi.ss.usermodel.Cell; import org.apache.poi.ss.usermodel.CellType; import org.apache.poi.ss.usermodel.Row; import org.apache.poi.ss.usermodel.Sheet; import org.apache.poi.ss.usermodel.Workbook; import org.apache.poi.xssf.usermodel.XSSFWorkbook; public class ExcelMonitor { public static void main(String[] args) { String filePath = "D:\\Xaltius(New)\\Material\\NTUC\\CNATIVE\\DevOps Introduction\\ScienceFair.xlsx"; // Path to the Excel file try (FileInputStream fis = new FileInputStream(filePath); Workbook workbook = new XSSFWorkbook(fis)) { Sheet sheet = workbook.getSheetAt(0); // Assume the first sheet boolean errorsFound = false; // Iterate over rows (skip the header row) for (int i = 1; i <= sheet.getLastRowNum(); i++) { Row row = sheet.getRow(i); if (row == null) continue; // Skip empty rows Cell nameCell = row.getCell(0); Cell projectCell = row.getCell(1); Cell judge1Cell = row.getCell(2); Cell judge2Cell = row.getCell(3); Cell avgCell = row.getCell(4); // Check for missing data if (isCellEmpty(nameCell) || isCellEmpty(projectCell)) { System.out.println("Row " + (i + 1) + ": Missing student name or project title."); errorsFound = true; } // Check score validity double judge1 = getCellNumericValue(judge1Cell); double judge2 = getCellNumericValue(judge2Cell);

```
            if (judge1 < 0 || judge1 > 10) {
                System.out.println("Row " + (i + 1) + ": Judge 1
score is out of range.");
                errorsFound = true;
            }
            if (judge2 < 0 || judge2 > 10) {
                System.out.println("Row " + (i + 1) + ": Judge 2
score is out of range.");
                errorsFound = true;
            }

            // Check average score calculation
            if (!isCellEmpty(avgCell) && judge1 >= 0 && judge2
>= 0) {
                double correctAvg = Math.round((judge1 + judge2)
/ 2 * 10.0) / 10.0;
                double avg = getCellNumericValue(avgCell);
                if (avg != correctAvg) {
                    System.out.println("Row " + (i + 1) + ": Incorrect
average score. Expected: " + correctAvg);
                    errorsFound = true;
                }
            }
        }

        if (!errorsFound) {
            System.out.println("All data is correct!");
        }

    } catch (IOException e) {
        System.err.println("Error reading the Excel file: " +
e.getMessage());
    }
}

private static boolean isCellEmpty(Cell cell) {
    return cell == null || cell.getCellType() == CellType.BLANK;
}

private static double getCellNumericValue(Cell cell) {
    if (cell == null || cell.getCellType() != CellType.NUMERIC) {
        return -1; // Return -1 to indicate invalid or missing
numeric data
    }
    return cell.getNumericCellValue();
}
}
```

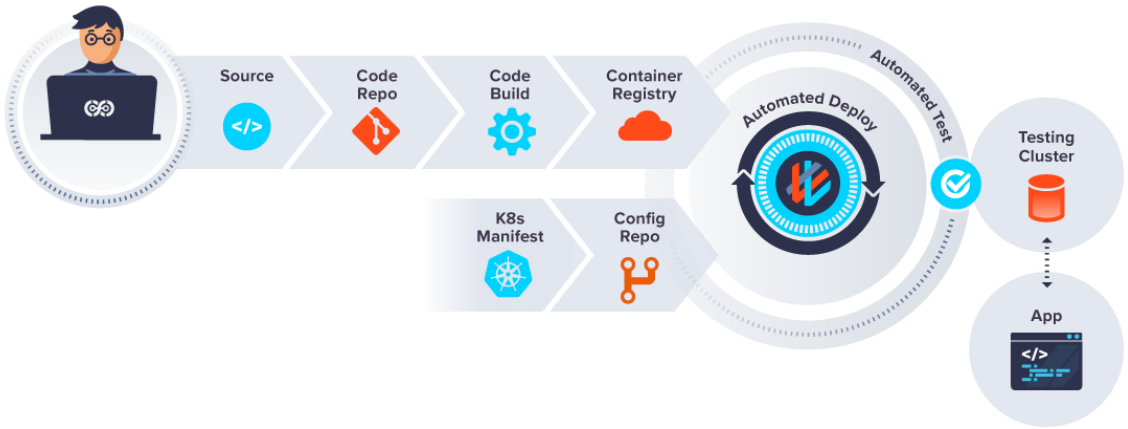
4. Save the provided Excel data (ScienceFair.xlsx) in the project folder.	
5. Compile Program	javac -cp poi-ooxml-5.2.3.jar:. ExcelMonitor.java
6. Run Program	java -cp poi-ooxml-5.2.3.jar:. ExcelMonitor
7. **Set Up a GitHub Repository** • Create a free GitHub account if you don't have one.	
8. Clone Repository by using git command.	
9. Add the "ExcelMonitor.java" and "ScienceFair.xlsx" file in repository.	Git add .
10. Commit the file with the comments in repository.	Git commit -m "Comment"

11. Push the file in repository.	Git push origin main
12. Now add the new folder in repository by using github browser. Follow the following steps.	
13. Create a file named “excel-monitor.yml”	
14. Paste the following code in “excel-monitor.yml” file.	name: Excel File Validator on: push: paths: - science_fair.xlsx jobs: validate_excel: runs-on: ubuntu-latest steps: - name: Checkout Code uses: actions/checkout@v3 - name: Set Up Java uses: actions/setup-java@v3 with: distribution: 'zulu' java-version: '17'

	- name: Install Apache POI Library run: | mkdir lib curl -L https://repo1.maven.org/maven2/org/apache/poi/poi/5.2.3/poi-5.2.3.jar -o lib/poi.jar curl -L https://repo1.maven.org/maven2/org/apache/poi/poi-ooxml/5.2.3/poi-ooxml-5.2.3.jar -o lib/poi-ooxml.jar - name: Compile Java Program run: javac -cp lib/poi.jar:lib/poi-ooxml.jar:. ExcelMonitor.java - name: Run Java Program run: java -cp lib/poi.jar:lib/poi-ooxml.jar:. ExcelMonitor
15. Check the file structure in github.	
16. Now click on “Action” button in github browser.	

Module 9: Future Trends and Innovations in DevOps Practices

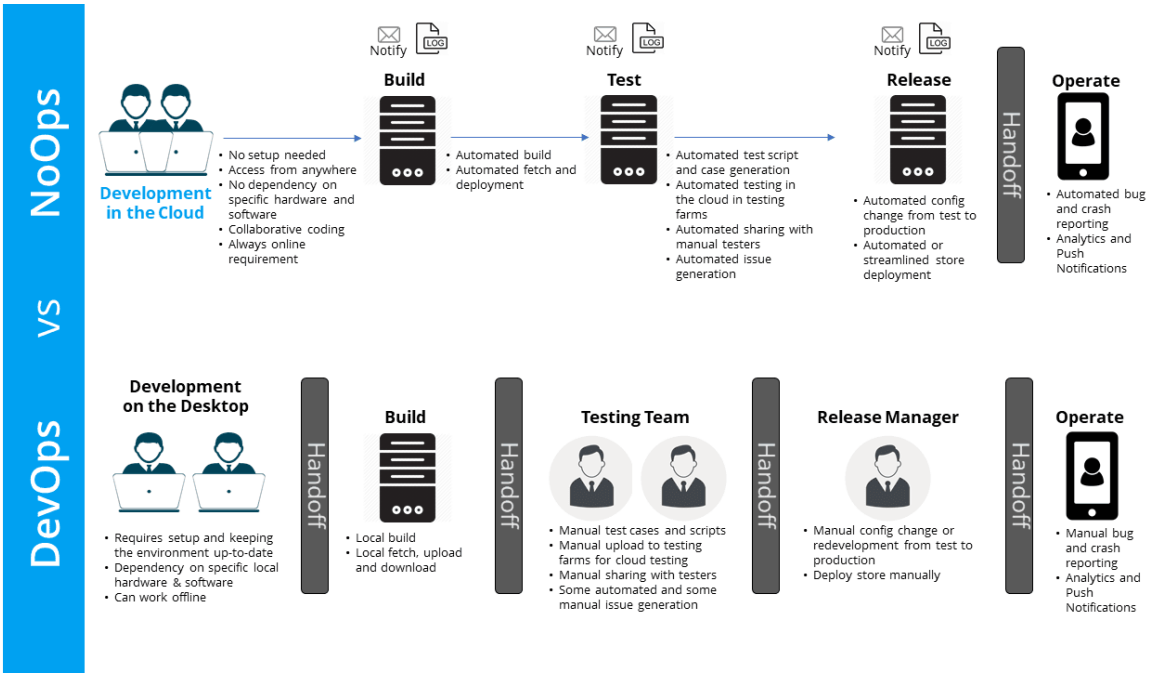
Exploring emerging trends in DevOps, including GitOps, NoOps, and AIOps:

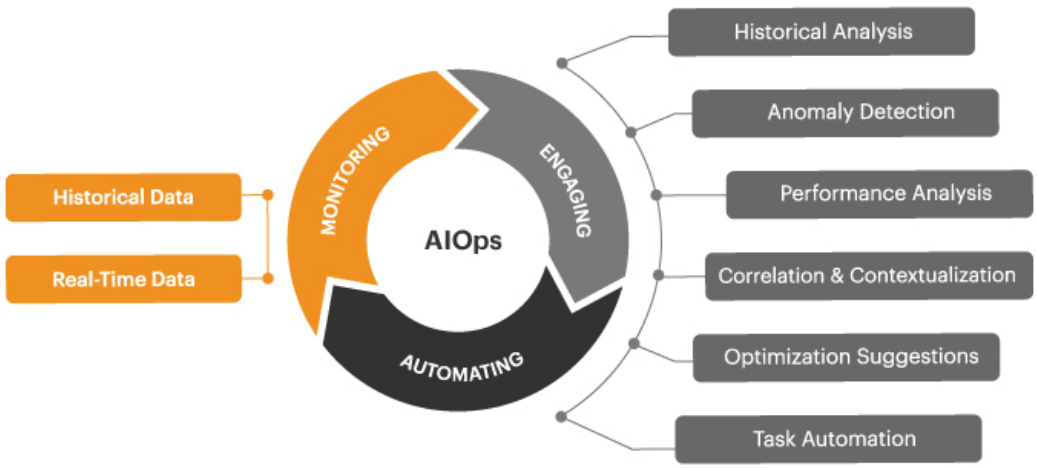


GitOps is a standardized approach to deploying, configuring, monitoring, updating, and managing infrastructure as code. GitOps keeps documentation, code, and any other information related to deployment in a version control system like Git, and makes updates automatically using automated directors.

NoOps is the concept that operations can be completely automated, with zero need for software management. NoOps follows a trend that has been in development for over a decade now.

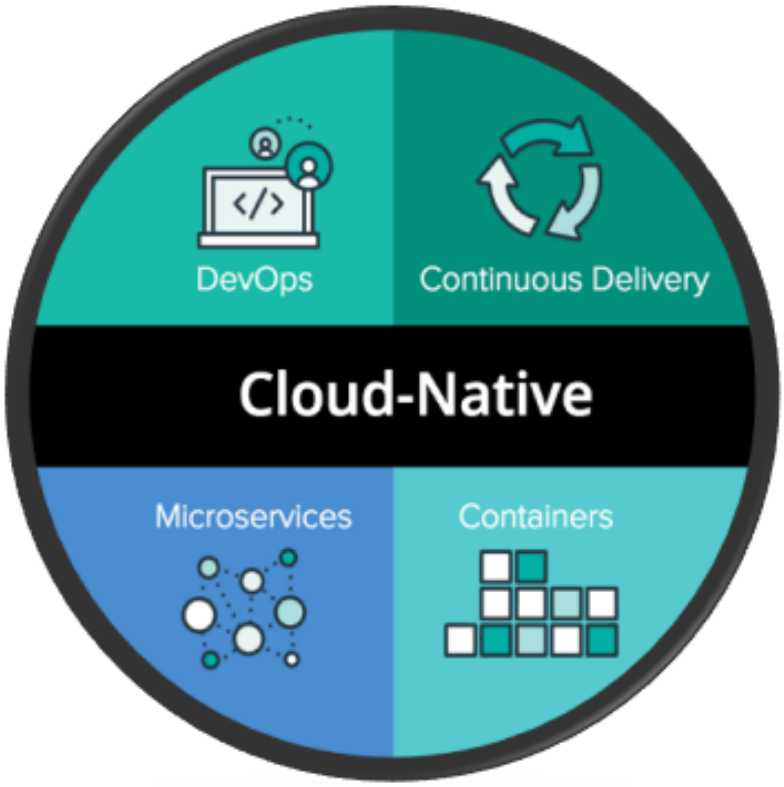
DevOps vs NoOps:





AIOps stands for artificial intelligence for IT operations. AIOps is a model that recognizes that advances in big data and machine learning are the keys to building new approaches in the rapidly evolving IT environment. The concept refers to a set of technologies that utilize analytics and machine learning (ML) to enhance and automate IT operations.

Recognizing the Influence of Cloud-Native Practices and AI in DevOps



- These applications are run and hosted in the cloud and are designed to capitalize on the inherent characteristics of a cloud computing software delivery model.
- Cloud-native applications use a microservice architecture. This architecture efficiently allocates resources to each service that the application uses, making the application flexible and adaptable to a cloud architecture.

Reflecting on the future prospects of DevOps practices.

How DevOps is Shaping the Future:

1. **Building Things Faster**:
    - In the future, everything will happen quickly. Apps on phones, games, or even robots will get updates instantly. DevOps will make this possible by helping teams work faster.
2. **Making Things Work Better**:
    - Imagine your favorite online game never crashes, and it always works perfectly. That's what DevOps aims to do for all kinds of technology.
3. **Helping Solve Big Problems**:
    - With DevOps, scientists and engineers can create tools to fight diseases, protect the environment, and even explore space more efficiently.

4. **Robots and AI**:
    - Robots and AI are like the super-smart helpers of the future. DevOps will make sure these helpers can learn and improve without any mistakes.

Fun Activities to Understand DevOps:

1. **Lego City Challenge**:
    - Divide into two teams: one to build (developers) and one to check if everything works (operations). Work together to complete the city fast and without any wobbly parts.

2. **Robot Race**:
    - Program a small robot (or pretend to) and make sure it can fix itself if it gets stuck. Talk about how teams make real robots smarter.

3. **Future Inventions Game**:
    - Imagine a new invention like a flying car or a talking treehouse. Discuss how DevOps could help make it real.

WITH OUR WEALTH OF INDUSTRY EXPERIENCE, NTUC LEARNINGHUB PROVIDES HIGH-QUALITY TRAINING PROGRAMMES THAT ARE INNOVATIVE, CONTEMPORARY AND AFFORDABLE.

These consist of programmes in:

- **Infocomm Technology**
- **Employability & Literacy**
- **Business Excellence**
- **Human Resources**
- **Healthcare**
- **Security**
- **Workplace Security & Health**
- **Foreign Worker Training**

For more information, please visit our website at **http://www.ntuclearninghub.com/** or call us at **6336-5482**