WEEK 1

Q.Write code for Simple user registration form for an event.

DevOps is the combination of Development + Operations, we focus on development aspect by creating a simple user registration form.This form serves as the foundational application and can be developed using various technologies such as:

Method	Description	Tools / Tecnologies
1.**Static Web page**	Simple HTML + CSS	VS Code, Browser
2.**With JavaScript**	Add validation and interactivity	JS, Bootstrap
3.**Using Python Flask**	Backend processing	Flask, Python, HTM
4.**Using Node.js + Express**	JavaScript-based server-side rendering	Node.js, Express
5.**Using React or Angular**	Modern frontend framework	ReactJS, Angular
6.**Using Django (Python)**	Full-stack web framework	Django
7.**PHP + MySQL**	Traditional stack	XAMPP/LAMP

We will go with method 3 - Python + Flask, since DevOps tools like Jenkins, Docker, Kubernetes, etc., are often used to deploy and test Flask apps.

Tools Needed:
- python 3.7+
- Flask (use pip install flask)
- Command Line Interface
- Browser

Implementation:
Step 1: Set up project structure

```
registration-form/
│
├── app.py
├── templates/
│   └── register.html
└── static/
└── style.css
```

Step 2: app.py – Main Flask Application

```
from flask import Flask, render_template, request, redirect, url_for
app = Flask(__name__)
@app.route('/', methods=['GET', 'POST'])
def home():
if request.method == 'POST':
print("Form submitted successfully!")
print("Form Data:", request.form)
return "Registration successful!"
return render_template('register.html')
if __name__ == '__main__':
app.run(debug=True)
```

Step 3: register.html – HTML Template in templates folder

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>User Registration</title>
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.
min.css" rel="stylesheet">
<style>
body {
background-color: #f7f7f7
}
form-container {
margin-top: 50px;
padding: 30px;
background: white;
border-radius: 10px;
box-shadow: 0 0 10px rgba(0,0,0,0.15);
}
</style>
</head>
<body>
<div class="container">
<div class="row justify-content-md-center">
<div class="col-md-6 form-container">
<h2 class="text-center mb-4">User Registration Form</h2>
<form method="POST" action="/"
<div class="mb-3">
```

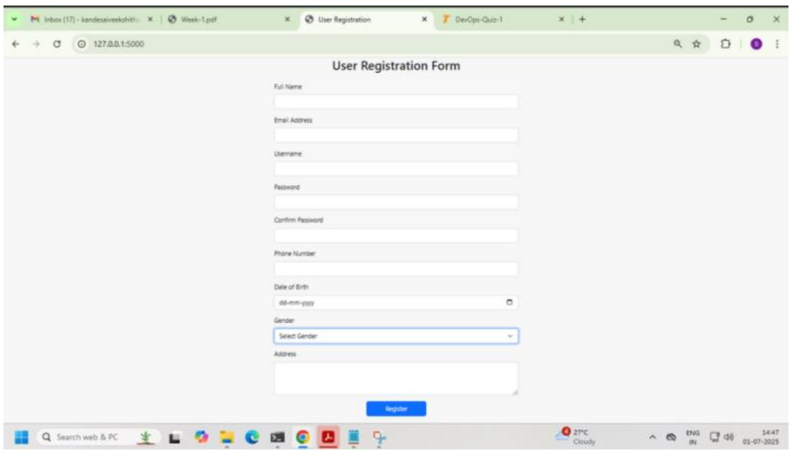
```
<label class="form-label">Full Name</label>
<input type="text" class="form-control" name="full_name">
</div>
<div class="mb-3">
<label class="form-label">Email Address</label>
<input type="email" class="form-control" name="email">
</div>
<div class="mb-3">
<label class="form-label">Username</label>
<input type="text" class="form-control" name="username">
</div>
<div class="mb-3">
<label class="form-label">Password</label>
<input type="password" class="form-control"
name="password">
</div>
<div class="mb-3">
<label class="form-label">Confirm Password</label>
<input type="password" class="form-control"
name="confirm_password">
</div>
<div class="mb-3">
<label class="form-label">Phone Number</label>
<input type="tel" class="form-control" name="phone">
</div>
<div class="mb-3">
<label class="form-label">Date of Birth</label>
<input type="date" class="form-control" name="dob">
</div>
<div class="mb-3">
<label class="form-label">Gender</label>
<select class="form-select" name="gender">
<option value="">Select Gender</option>
<option>Male</option>
<option>Female</option>
<option>Other</option>
</select>
</div>
<div class="mb-3">
<label class="form-label">Address</label>
<textarea class="form-control" rows="3”
name="address"></textarea>
</div>
<div class="text-center">
<button type="submit" class="btn btn-primary px-
5">Register</button>
```

```
</div>
</form>
</div>
</div>
</div>
</body>
</html>
```

To run the above program:

- ➢ pip install flask
- ➢ python app.py
- ➢ Visit: http://127.0.0.1:5000/ in your browser

OUTPUT



User Registration Form

Full Name

Email Address

Username

Password

Confirm Password

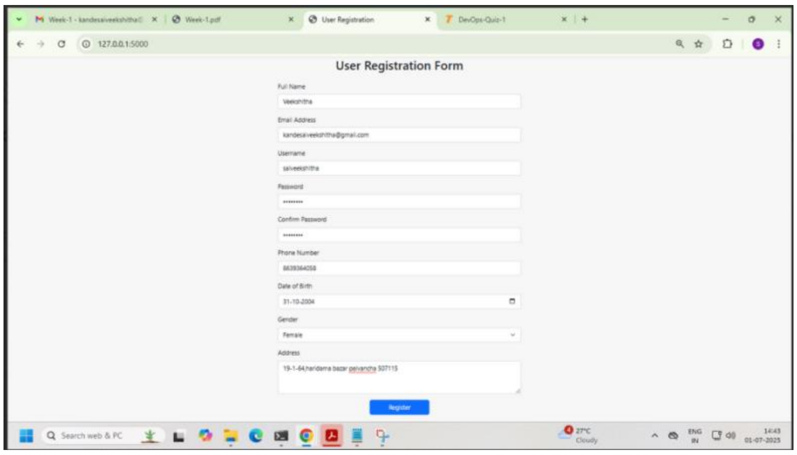
Phone Number

Date of Birth
dd-mm-yyyy

Gender
Select Gender

Address

Register



User Registration Form

Full Name
Veekshitha

Email Address
kandesaiveekshitha@gmail.com

Username
saiveekshitha

Password
••••••••

Confirm Password
••••••••

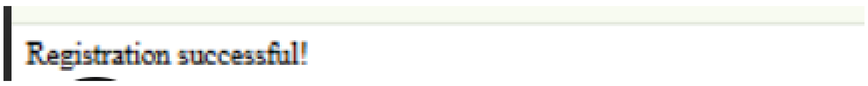
Phone Number
8639364058

Date of Birth
31-10-2004

Gender
Female

Address
19-1-64,haridama bazar palvancha 507115

Register



Registration successful!

WEEK 2

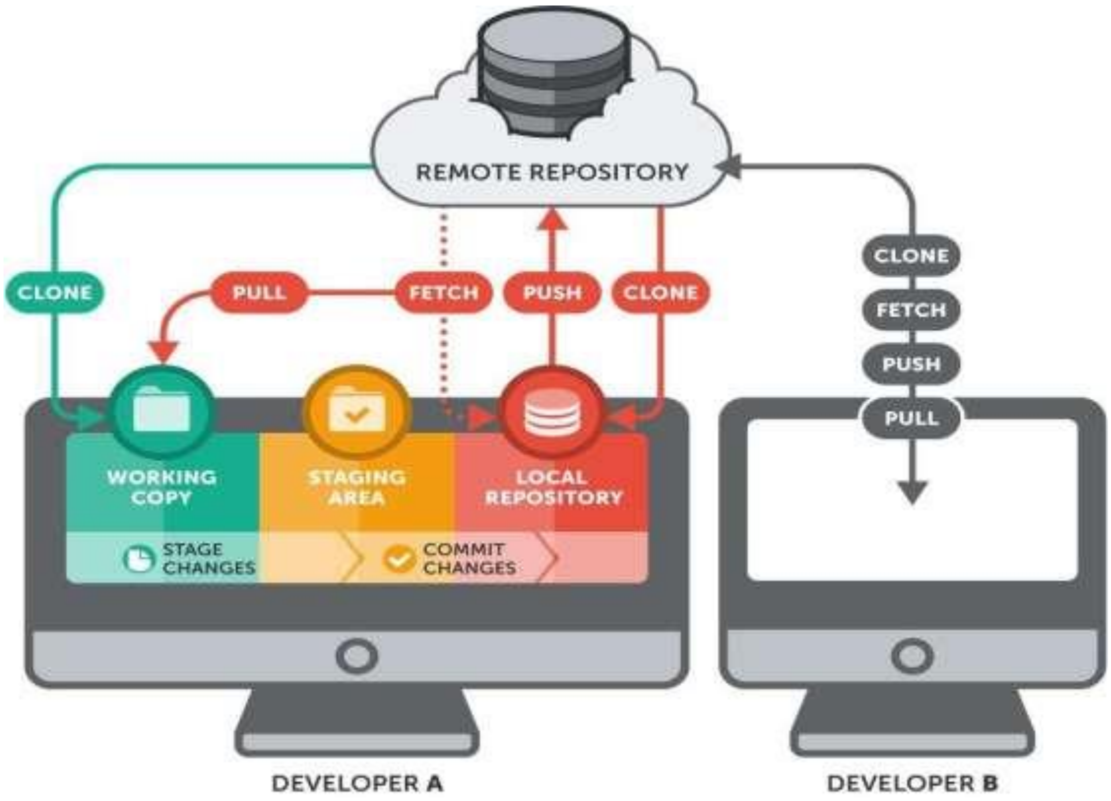
Q.Explore Git and GitHub commands

A version control system (VCS) is a software tool that helps manage changes to source code over time. It allows multiple developers to collaborate on a project, track changes, and revert to previous versions when needed. Version control systems are essential in software development to maintain code integrity, facilitate teamwork, and manage the complexity of evolving projects

Git is one of the most popular distributed version control systems. Introduced by Linus Torvalds in 2005, Git provides a distributed and decentralized model where each developer has a complete copy (clone) of the entire repository on their local machine. This allows developers to work offline, commit changes locally, and synchronize their work with others using remote repositories. **GitHub** is a web based hosting service for Git repositories.

Why Git in DevOps ?

- Supports Continuous Integration (CI) workflows
- Enables collaboration and rollback
- Integrates with build and deployment tools (Jenkins, Docker)



1. Working Directory

Definition: The working directory is where you make changes to your files. It contains the files you are currently editing and working on.
Example: If you have a project folder on your computer with files like register.html, sample.txt, the folder and its contents are your working directory.

2. Staging Area

Definition: The staging area (also known as the index) is a place where you prepare changes before committing them.
Example: Suppose you modified register.html and sample.txt. If you only want to commit changes made to register.html for now, you would add register.html to the staging area using git add register.html. The changes to sample.txt remain in the working directory but are not yet staged.

3. Local Repository

Definition: The local repository is where Git stores your commits, branches, and history on your local machine. It's the place where you commit your changes and manage your project.
Example: After staging and committing changes, the Git records are stored in your local repository. You can check the history of your commits with git log, and your commits are saved in your local.git directory

4. Remote Repository

Definition: The remote repository is a version of your project stored on a server or a platform like GitHub, GitLab, or Bitbucket. It allows you to share and collaborate on your project with others.
Example: If you push your local commits to a repository on GitHub using git push origin master, you are updating the remote repository at https://github.com/username/repository.git with your local changes.

1. Install Git: Download and install Git for Windows from the official site:

https://git-Scm.com/downloads/win

- During installation: Choose "Use Git from the command line"
- Keep other default settings,
- After installation, open Command Prompt (CMD) or Git Bash

2. Create a GitHub Account:

Sign up at: https://github.com
Use your official/student email ID.

3. Create or Navigate to a Project Folder

Navigate to your application folder
C:\Users\CC5\Documents\12C4\

4. Configure Git (First Time Only)

These commands set your name and email for commits:

git config --global user.name "Your Name"

```
C:\Users\Admin\Documents\22251A12C3\Userregistration>git config --global user.name "veekshitha"
```

git config --global user.email "your-email@example.com"

```
C:\Users\Admin\Documents\22251A12C3\Userregistration>git config --global user.email
"kandesaiveekshitha@gmail.com"
```

--global: applies to all projects on your system
--local: applies only to the current project folder

5. Verify configuration:

git config --list

You should see your name and email listed.

```
C:\Users\Admin\Documents\22251A12C3\Userregistration>git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=true
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=main
core.editor="C:\Users\Admin\AppData\Local\Programs\Microsoft VS Code\bin\code" --wait
user.email=kandesaiveekshitha@gmail.com
user.name=veekshitha
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.ignorecase=true
```

6. Initialize a Local Git Repository

git init

This creates a hidden .git folder. Git will now track changes in this folder.

```
C:\Users\Admin\Documents\22251A12C3\Userregistration>git init
Reinitialized existing Git repository in C:/Users/Admin/Documents/22251A12C3/Userregistration/.git/
```

7. Check File Status

Shows untracked, modified, or staged files.

```
C:\Users\Admin\Documents\22251A12C3\Userregistration>git status
On branch main
No commits yet
```

8. Stage Files for Commit

Selecting the changed files you want to include in your next commit.

git add app.py # add a specific file
git add . # add all files

```
C:\Users\Admin\Documents\22251A12C3\Userregistration>git commit -m "Committed all files"
[main (root-commit) bc874ef] Committed all files
 1 file changed, 11 insertions(+)
 create mode 100644 app.py
```

9. Commit Changes

git commit -m "Committed all files"
saving a snapshot of your code with a message

10.Connect to Remote GitHub Repository

Step 1: Create a GitHub Repo
Example: https://github.com/sriludone/veekshitha.git
Step 2: Add the remote

```
C:\Users\Admin\Documents\22251A12C3\Userregistration>git remote add origin
https://github.com/saiveekshitha/App.git
```

Step 3: Verify remote

```
git remote -v
C:\Users\Admin\Documents\22251A12C3\Userregistration>git remote -v
origin  https://github.com/saiveekshitha/App.git (fetch)
origin  https://github.com/saiveekshitha/App.git (push)
```

11. Push Code to GitHub

First Push:
git push -u origin master
This command uploads your local master branch to the remote repository (GitHub) and sets a tracking link so you can use git push next time without full details.
-u means --set-upstream
It tells Git to remember the connection between your local branch and the remote branch
Later pushes:

```
git push
C:\Users\Admin\Documents\22251A12C3\Userregistration>git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 6 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 454 bytes | 454.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/saiveekshitha/App.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

12. Pull Latest Changes from GitHub

git pull
Fetches the latest changes from the remote repository and merges them into your current local branch.

13. Fetch Changes

git fetch
downloads the latest changes from the remote repository, but does not apply (merge) them into your code

14. Branching in Git: A branch in Git is like a separate version of your project. It allows you to make changes or add new features without affecting the main version of your project.
Main Branch (Master): This is your original branch with all the documents you currently have.
New Branch: You create a new branch to add your new files. This branch is like a separate copy of your book where you work on the new chapter.

- View branches:

git branch
shows all the branches in your local Git repository.
- Create a new branch:

git branch new
- Switch between branches:

git checkout main
- Merge branch into current:

git merge new

15. Delete or Rename Files

- Delete from Git:

git rm filename.txt
- Rename file:

git mv oldname.txt newname.txt

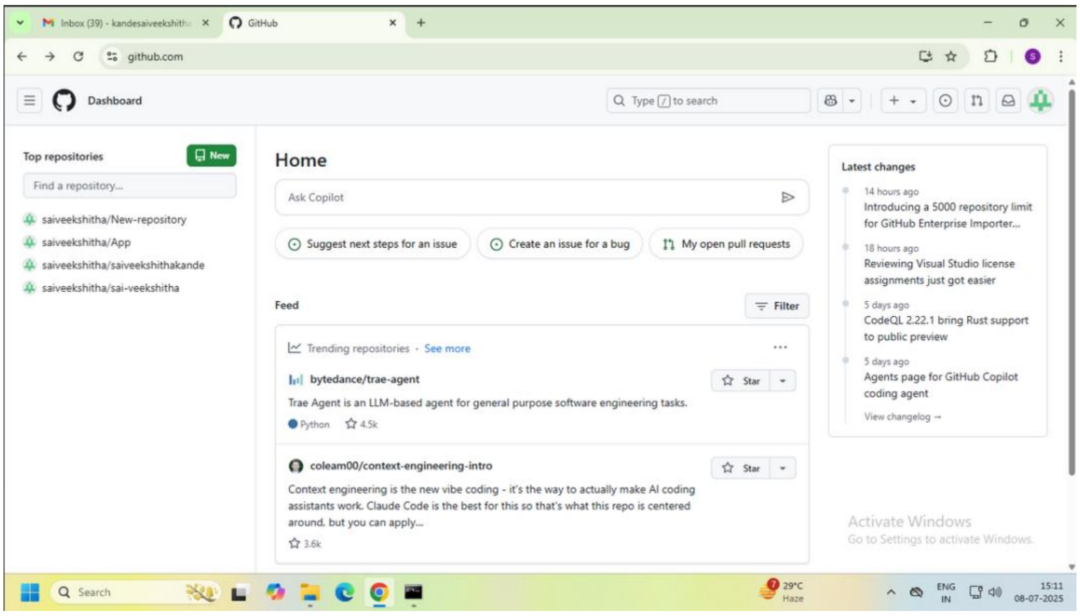
16. Get Help for Any Command

git help <command>
git help commit
git help push

17. Clone an Existing Repository (From GitHub)

git clone https://github.com/sriludone/app.git
This creates a full local copy of the GitHub repository

OUTPUT


Inbox (39) - kandesaiveekshitha
GitHub
github.com
Dashboard
Type / to search
Top repositories
New
Find a repository...
saiveekshitha/New-repository
saiveekshitha/App
saiveekshitha/saiveekshithakande
saiveekshitha/sai-veekshitha
Home
Ask Copilot
Suggest next steps for an issue
Create an issue for a bug
My open pull requests
Feed
Filter
Trending repositories · See more
bytedance/trae-agent
Star
Python
4.5k
coleam00/context-engineering-intro
3.6k
Latest changes
View changelog →
Activate Windows
Go to Settings to activate Windows.
Search
29°C Haze
ENG IN
15:11
08-07-2025

Trae Agent is an LLM-based agent for general purpose software engineering tasks.

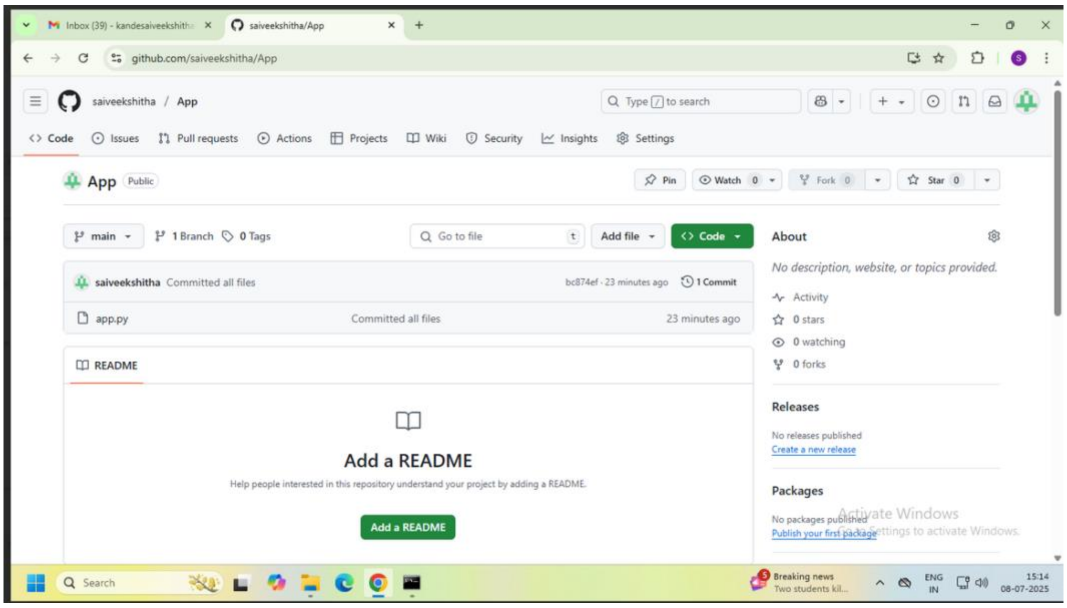
Context engineering is the new vibe coding - it's the way to actually make AI coding assistants work. Claude Code is the best for this so that's what this repo is centered around, but you can apply...

14 hours ago
Introducing a 5000 repository limit for GitHub Enterprise Importer...

18 hours ago
Reviewing Visual Studio license assignments just got easier

5 days ago
CodeQL 2.22.1 bring Rust support to public preview

5 days ago
Agents page for GitHub Copilot coding agent



saiveekshitha Committed all files bc874ef · 23 minutes ago 1 Commit

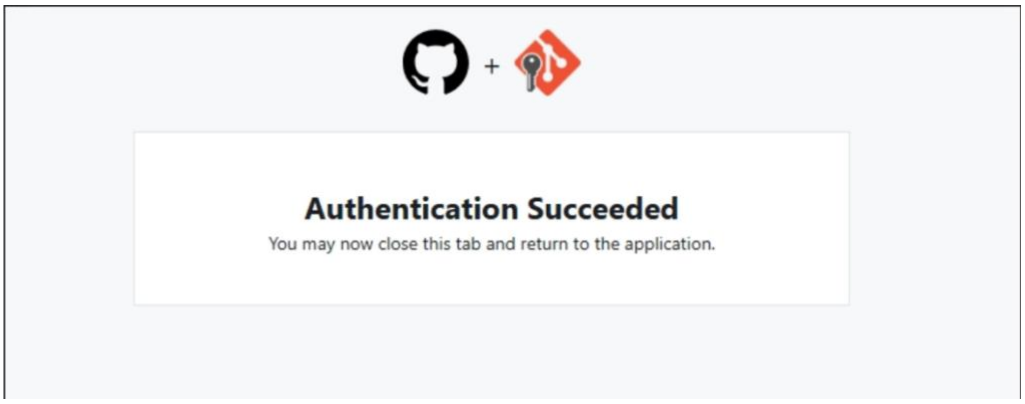
app.py Committed all files 23 minutes ago

Help people interested in this repository understand your project by adding a README.

No description, website, or topics provided.
Activity
0 stars
0 watching
0 forks

No releases published
Create a new release

No packages published
Publish your first package



Authentication Succeeded

You may now close this tab and return to the application.

WEEK 3:

Q. Practice Source code management on GitHub.Exeriment with the source code written in exercise 1

DevOps emphasizes collaboration, automation, and continuous integration. A core DevOps practice is Source Code Management (SCM) — storing, tracking, and collaborating on code via repositories.

In Week 1, we developed a basic application (user registration form).

In Week 2, we learned fundamental Git/GitHub commands.

In this Week 3, we take it further by:

- Hosting the code on GitHub
- Managing branches
- Tracking changes
- Collaborating using pull and push
- Understanding SCM best practices

❖ **Step 1: Create a Repository on GitHub**

- Go to https://github.com
- Click New Repository
- Enter Repository name: Registration
- Description: “Simple Event Registration App”
- Keep it Public
- Do not initialize with README
- Click Create repository

❖ **Step 2: Connect Local Project to GitHub**

- Open terminal or Git Bash
- Navigate to your project directory:cd path/to/registration-form
- Initialize git (if not already done):git init
- Add your project files: git add .
- Commit changes: git commit -m "Initial commit - Registration form UI"
- Add the GitHub repository as remote: git remote add origin https://github.com/yourusername/registration-form.git
- Push your code: git push -u origin master

❖ **Step 3: Create a New Feature Branch**

Next we will simulate feature development using branching:

git branch add-validation
git checkout add-validation

Now you are on a new branch add-validation.

- **Step 4: Make Changes**
  Edit your Flask or HTML form to add validations (e.g., phone number length or email format). Save changes.
- **Step 5: Stage, Commit, and Push Branch**
  - git add .
  - git commit -m "Add phone and email validation to form"
  - git push origin add-validation

- **Step 6: Create Pull Request on GitHub**
  - Go to your repository on GitHub
  - GitHub will detect the pushed branch and show a Compare & pull request button - click it.
  - Add a meaningful description and title.
  - Click Create Pull Request
  - Click Merge Pull Request
  - Confirm by clicking Confirm Merge
  - (Optional) Delete the branch on GitHub

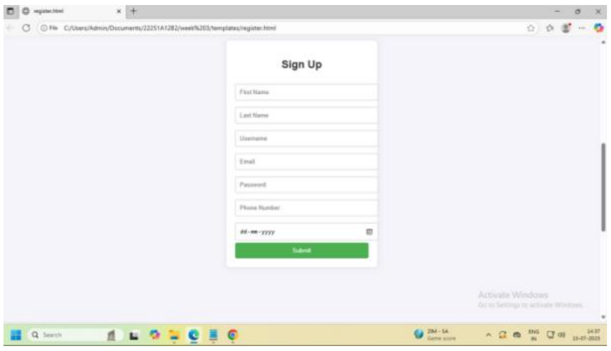
- **Step 7: Pull the Latest Changes Locally**
  - git checkout master
  - git pull origin master

```
<!Doctype html>
<html>
<head>
   <style>
      body {
         font-family: Arial, sans-serif;
         display: flex;
         justify-content: center;
         align-items: center;
         height: 300vh;
         background-color: #f4f4f9;
         margin: 0;
      }
      .form-container {
         background-color: #fff;
         padding: 20px;
         border-radius: 8px;
         box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
         width: 300px;
      }
      h2 {
         text-align: center;
         color: #333;
      }
      input {
         width: 100%;
         padding: 10px;
         margin: 10px 0;
         border: 1px solid #ccc;
         border-radius: 4px;
      }
      button {
         width: 100%;
         padding: 10px;
         background-color: #4CAF50;
         color: white;
         border: none;
         border-radius: 4px;
         cursor: pointer;
      }
      button:hover {
         background-color: #45a049;
      }
.alert {
         color: red;
         font-size: 12px;
margin-top: -5px;
 </style>
</head>
<body>
```

```
<div class="form-container">
  <h2>Sign Up</h2>
  <form id="myForm">
    <input type="text" id="firstName" placeholder="First Name" required>
    <div id="firstName-error" class="alert"></div>
    <input type="text" id="lastName" placeholder="Last Name" required>
    <div id="lastName-error" class="alert"></div>
    <input type="text" id="username" placeholder="Username" required>
    <div id="username-error" class="alert"></div>
    <input type="email" id="email" placeholder="Email" required>
    <div id="email-error" class="alert"></div>
    <input type="password" id="password" placeholder="Password" required>
    <div id="password-error" class="alert"></div>
    <input type="tel" id="phone" placeholder="Phone Number" required>
    <div id="phone-error" class="alert"></div>
    <input type="date" id="dob" placeholder="Date of Birth" required>
    <div id="dob-error" class="alert"></div>
    <button type="submit">Submit</button>
  </form>
</div>
<script>
  const form = document.getElementById("myForm");
  form.addEventListener("submit", function (e) {
    e.preventDefault();
    const fName = document.getElementById("firstName").value;
    const lName = document.getElementById("lastName").value;
    const user = document.getElementById("username").value;
    const email = document.getElementById("email").value;
    const pass = document.getElementById("password").value;

 const phone = document.getElementById("phone").value;
 const dob = document.getElementById("dob").value;

 let isValid = true;
 document.getElementById("firstName-error").textContent = "";
 document.getElementById("lastName-error").textContent = "";
 document.getElementById("username-error").textContent = "";
 document.getElementById("email-error").textContent = "";
 document.getElementById("password-error").textContent = "";
 document.getElementById("phone-error").textContent = "";
 document.getElementById("dob-error").textContent = "";
 const nameRegex = /^[a-zA-Z]{1,30}$/;
  if (!nameRegex.test(fName)) {
    document.getElementById("firstName-error").textContent =
      "First name must be 1-30 letters.";
    isValid = false;

  }
```

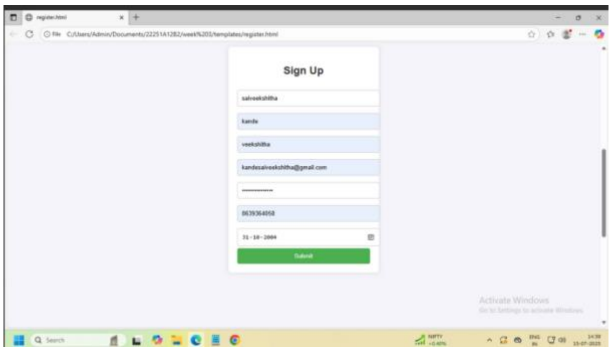
```
          if (!nameRegex.test(lName)) {
            document.getElementById("lastName-error").textContent =
              "Last name must be 1-30 letters.";
            isValid = false;
          }
          const userRegex = /^[a-zA-Z0-9_]{3,15}$/;
          if (!userRegex.test(user)) {
            document.getElementById("username-error").textContent =
              "Username must be 3-15 chars (letters, numbers, underscores).";
            isValid = false;
          }
          const emailRegex = /^([a-zA-Z0-9._%+-]+)@([a-zA-Z0-9.-]+)\.([a-zA-Z]{2,})$/;
          if (!emailRegex.test(email)) {
            document.getElementById("email-error").textContent =
              "Invalid email address.";
            isValid = false;
          }
          const passRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%?&])[A-Za-z\d@$!%?&]{8,}$/;
          if (!passRegex.test(pass)) {
            document.getElementById("password-error").textContent =
              "Password must have 8+ chars, 1 uppercase, 1 number, 1 special char.";


                        isValid = false;

                        }
          const phoneRegex = /^[7-9][0-9]{9}$/;
          if (!phoneRegex.test(phone)) {
            document.getElementById("phone-error").textContent =
              "Phone must be 10 digits, starting with 7-9.";
            isValid = false;
          }
          const today = new Date();
          const bDate = new Date(dob);
          const age = today.getFullYear() - bDate.getFullYear();
          const month = today.getMonth() - bDate.getMonth();
                if (isValid) {
            alert("Form submitted successfully!");
          }
        });
    </script>
</body>
</html>
```

OUTPUT



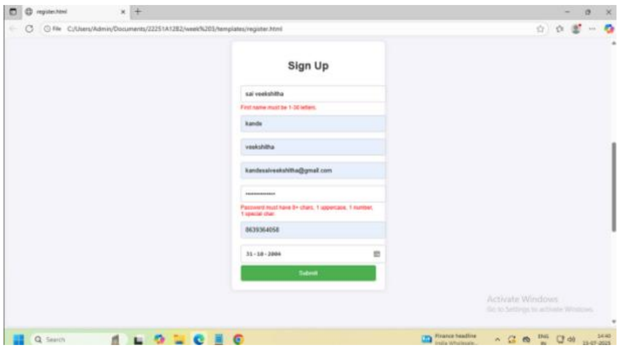
Sign Up

First Name
Last Name
Username
Email
Password
Phone Number
dd-mm-yyyy
Submit



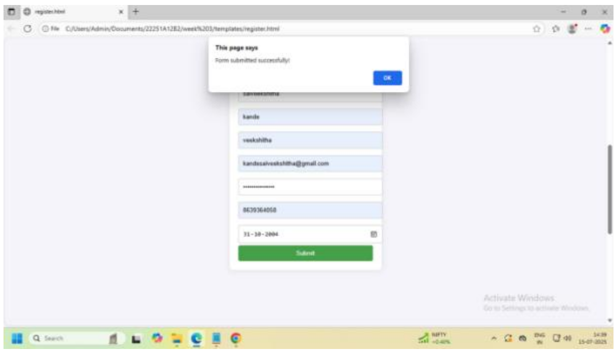
Sign Up

saiveekshitha
kande
veekshitha
kandesaiveekshitha@gmail.com
••••••••••••••
8639364058
31-10-2004
Submit



Sign Up

sai veekshitha
First name must be 1-30 letters.
kande
veekshitha
kandesaiveekshitha@gmail.com
••••••••••••••
Password must have 8+ chars, 1 uppercase, 1 number, 1 special char.
8639364058
31-10-2004
Submit

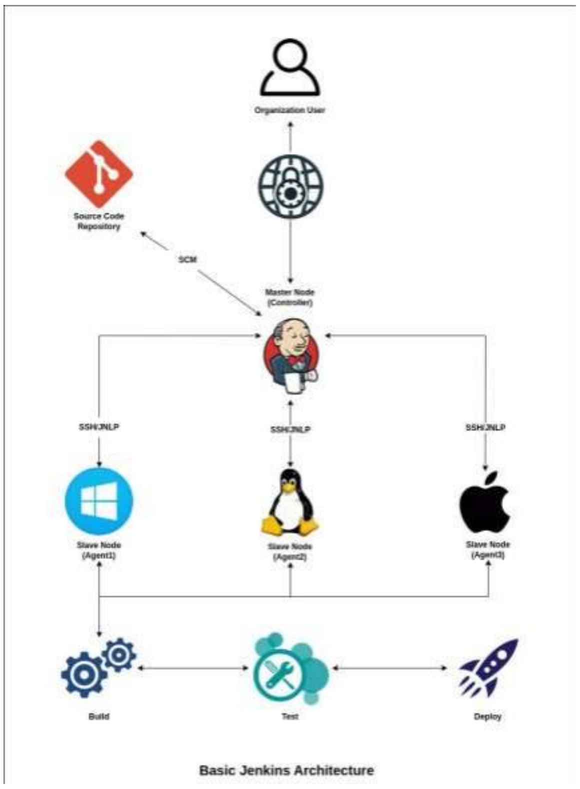


This page says
Form submitted successfully!
OK

saiveekshitha
kande
veekshitha
kandesaiveekshitha@gmail.com
••••••••••••••
8639364058
31-10-2004
Submit

WEEK 4

Jenkins installation and setup,explore the environment.
Jenkins is an **open-source automation server** widely used in DevOps pipelines for **Continuous Integration (CI)** - automatically building and testing code each time a team member commits changes.
Jenkins was originally developed in **2004** as a project called **Hudson** by Sun Microsystems.
Due to some trademark issues after Oracle acquired Sun, the project was forked
and renamed **Jenkins** in **2011**. Since then, Jenkins has become one of the most widely used CI/CD tools globally, supported by a vibrant open-source community.



In Week 3, we hosted and managed source code using GitHub. This week, we integrate Jenkins with version control to automate build processes, enabling smooth progression to **CI/CD** in Week 4.
Jenkins **itself is written in Java**, so it runs on the Java Virtual Machine (JVM).

But it can **build, test, and deploy projects written in any language**, including:

- Java
- Python
- C/C++
- JavaScript (Node.js, Angular, React, etc.)
- PHP
- Ruby
- .NET (C#)
- Go, and more.

Pre-requisites:

- Java installed
- Git installed (to integrate with repositories later)

Step-by-Step Jenkins Installation (on Windows)

➢ **Step 1: Install Java**

Visit https://adoptium.net/en-GB/temurin/releases/
Download Java 21
Install and set up the environment variable: **java –version**

➢ **Step 2: Download Jenkins**

Visit https://www.jenkins.io/download/
Choose Windows → download .war file
Run the installer
Jenkins installs as a Windows service
copy jenkins.war file c drive then
cd C:\jenkins

➢ **Step 3: Run Jenkins:**

```
java -jar jenkins.war
```

Jenkins will start a built-in web server (usually at http://localhost:8080)
It will generate a default **admin password**
Then, you'll go to the browser → paste the password → continue setup.

➢ **Step 4: Customize Jenkins**

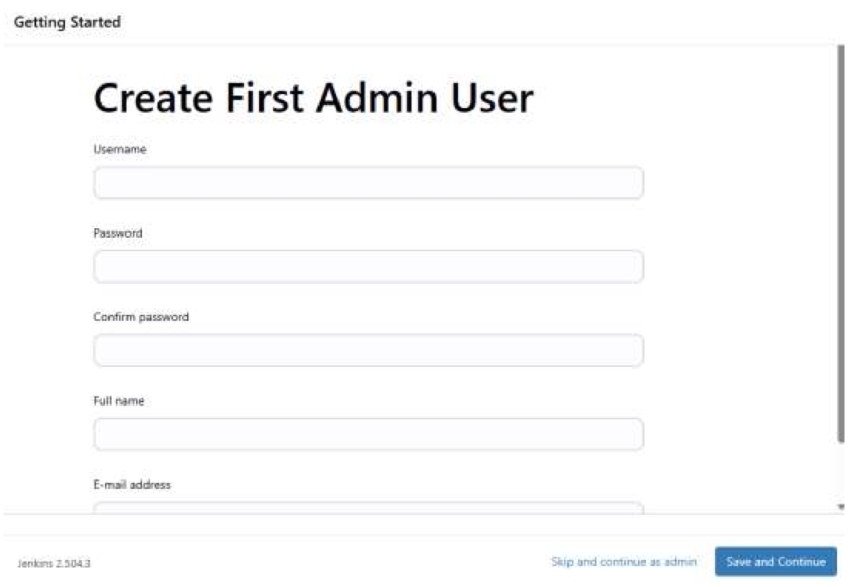
You'll get two options:

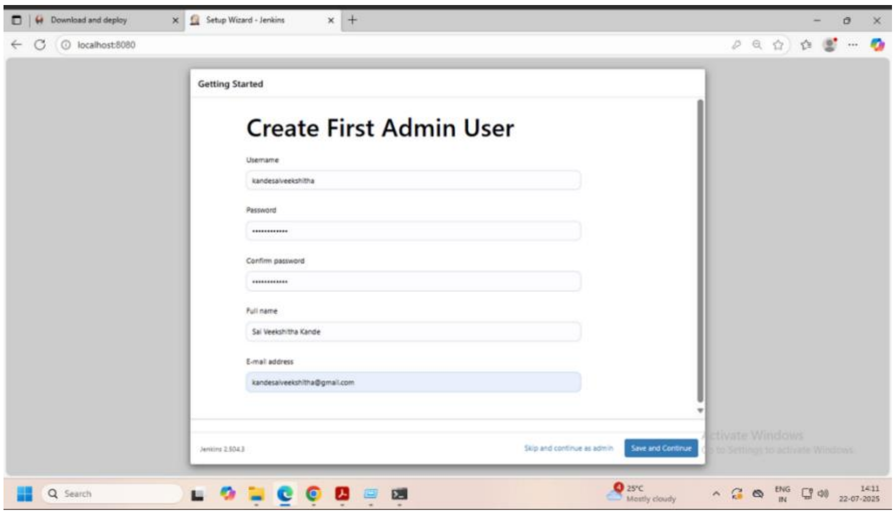
- Install suggested plugins (Recommended – choose this)
- Select plugins to install (Only if you know what you need)

Click "Install suggested plugins". It will download and install plugins like Git, Maven, Pipeline, etc.

➢ **Step 5: Create Admin User**

After plugins are installed, you'll be prompted to create an admin user:





Getting Started

Create First Admin User

Username: kandesaiveekshitha

Password: ••••••••••••

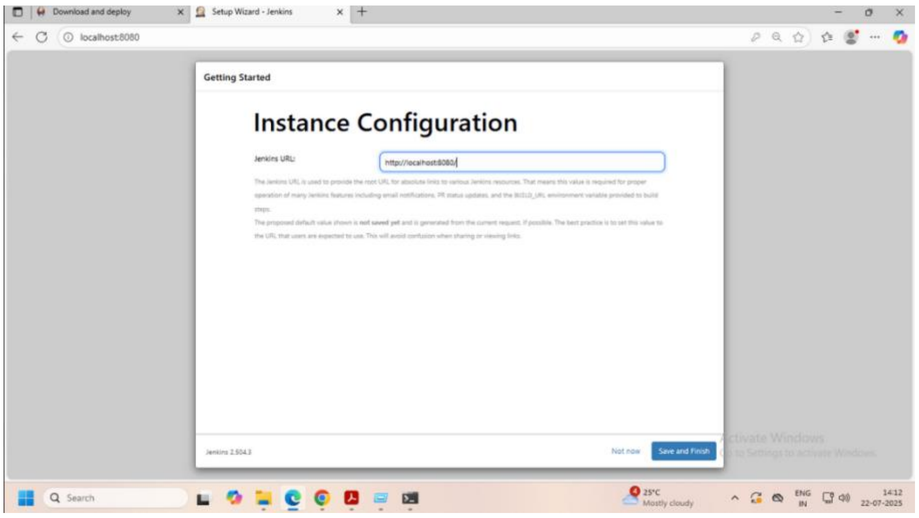
Confirm password: ••••••••••••

Full name: Sai Veekshitha Kande

E-mail address: kandesaiveekshitha@gmail.com

Jenkins 2.504.3

Step 5: Instance Configuration



Getting Started

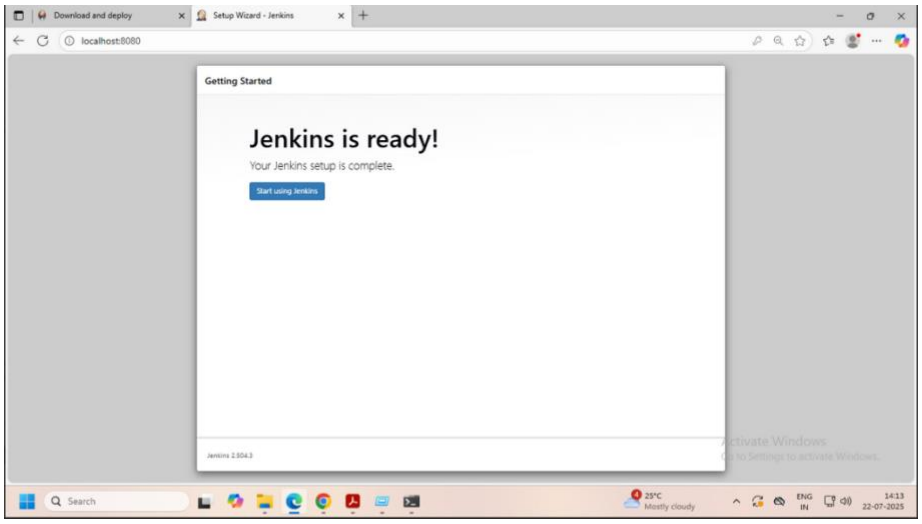
Instance Configuration

Jenkins URL: http://localhost:8080/

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.504.3

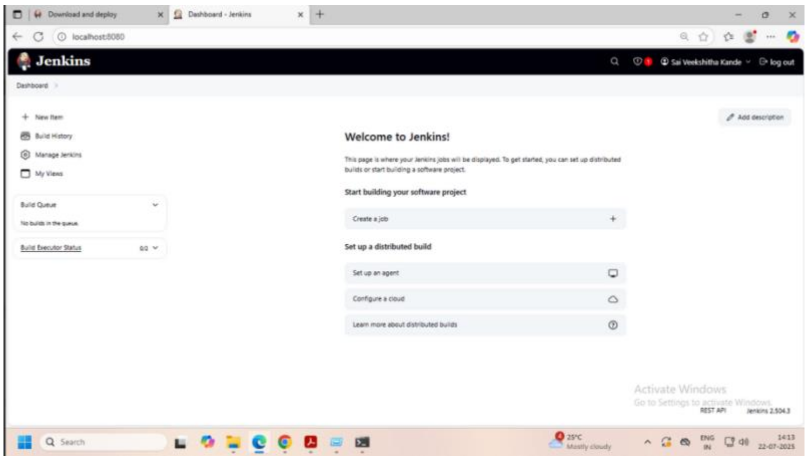


Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

Jenkins 2.504.3


Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Set up a distributed build

Explore the Environment:

● **New Item** – to create a job (Freestyle, Pipeline, etc.)
● **Build History** – view previous builds.
● **Manage Jenkins** – configure system-wide settings.
● **My Views** – customize job views.
● **Create a job** – set up Freestyle or Pipeline builds.

➢ **Manage Jenkins**

o **System**: Global settings such as Jenkins home, system email, and paths.
o **Tools**: Checked JDK 21 is detected (Temurin). Git and Maven can also be configured.
o **Plugins**: Viewed installed plugins. Jenkins allows managing plugins that extend functionality.
o **Security**: Saw how Jenkins manages users and permissions (important for CI/CD security).
o **Credentials**: Credentials like SSH keys, usernames/passwords can be stored securely and accessed by jobs.
o **Nodes**: This section allows adding distributed build agents (not configured in this lab).
o **Appearance**: Optional UI customization features.

WEEK 5

Q.Demonstrate continuous integration and development using Jenkins.

This week, we're setting up GitHub and Jenkins to work together so that whenever a new commit is pushed, Jenkins will automatically pull, build, and test the application. This process is called **Continuous Integration (CI)** — the practice of frequently merging code into a shared repository multiple times a day, followed by automated builds and tests to catch issues early.

Benefits:

- Detect integration issues early
- Fast feedback for developers
- Less manual work

Implementation:

Step 1: Create a Freestyle Jenkins Job

1. Go to **Jenkins dashboard**
2. Click **"New Item"**
3. Enter name: Week5-CICD
4. Select **"Freestyle project".** we have 5 types of projects here

Project Type	Description	Best For
Freestyle Project	Click-and-create job. Easy setup using buttons.	Beginners and small tasks
Pipeline	Job written as code (Jenkinsfile).	Big projects, full automation
Multiconfiguration	Test the same project in many ways (like on Windows & Linux).	Checking different setups
Folder / Org Folder	Group jobs in folders. Can auto-detect GitHub repos.	Large teams, organizing many jobs
Multibranch Pipeline	Jenkins creates jobs for every Git branch (if Jenkinsfile is there).	Teams using branches and pull requests

5. Configure GitHub Repository

Scroll to Source Code Management

Select Git

Paste your GitHub repo URL
Example: https://github.com/sriludone/Week-2.git

6.Build Triggers:
Poll SCM:
10 * * * *

7.Scroll to the Build section

Click Add build step
Choose: Execute Windows batch command
Add a text
echo Hello Jenkins from Windows!
Dir
Save and Build
Click Save

8.See the Output

Click the build number (#2, #3, etc.)

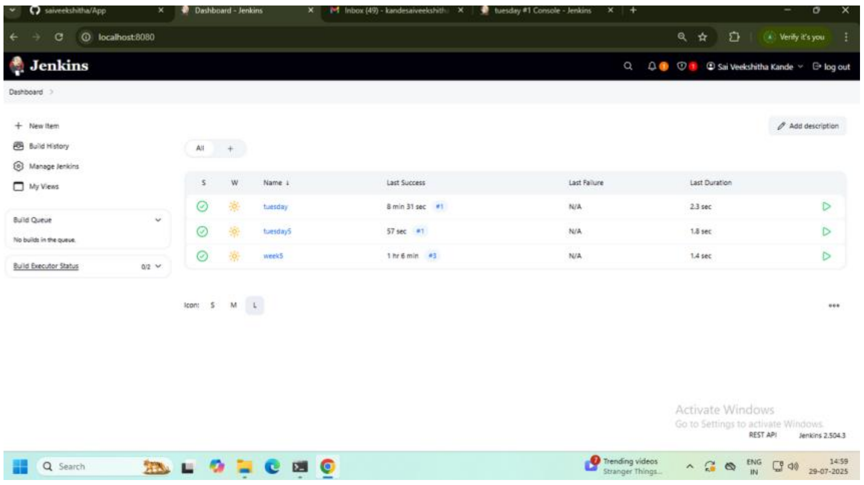
Click Console Output
Step 3: For Testing

- Open your local repo folder.
- Create a new file (new.txt)
- git add new.txt
- git commit -m "test file"
- git push –u origin master

Use this if getting authentication problems
git push https://github.com/USERNAME/REPO.git

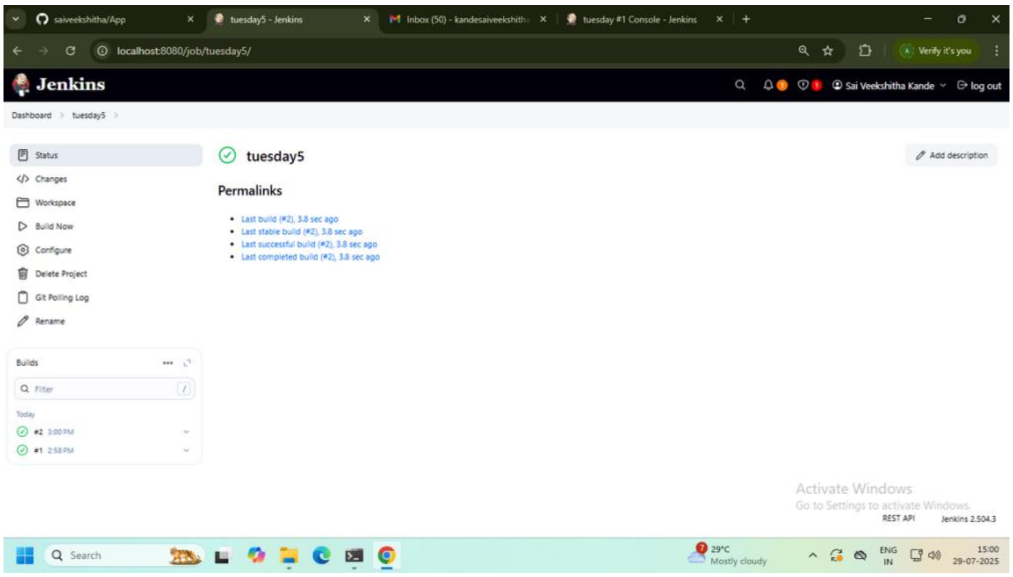
Go to Jenkins – you'll see the job starts automatically!

OUTPUT



Build Queue
No builds in the queue.
Build Executor Status 0/2

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀	tuesday	8 min 31 sec #1	N/A	2.3 sec
✓	☀	tuesday5	57 sec #1	N/A	1.8 sec
✓	☀	week5	1 hr 6 min #3	N/A	1.4 sec



localhost:8080/job/tuesday5/

Dashboard > tuesday5 >

tuesday5

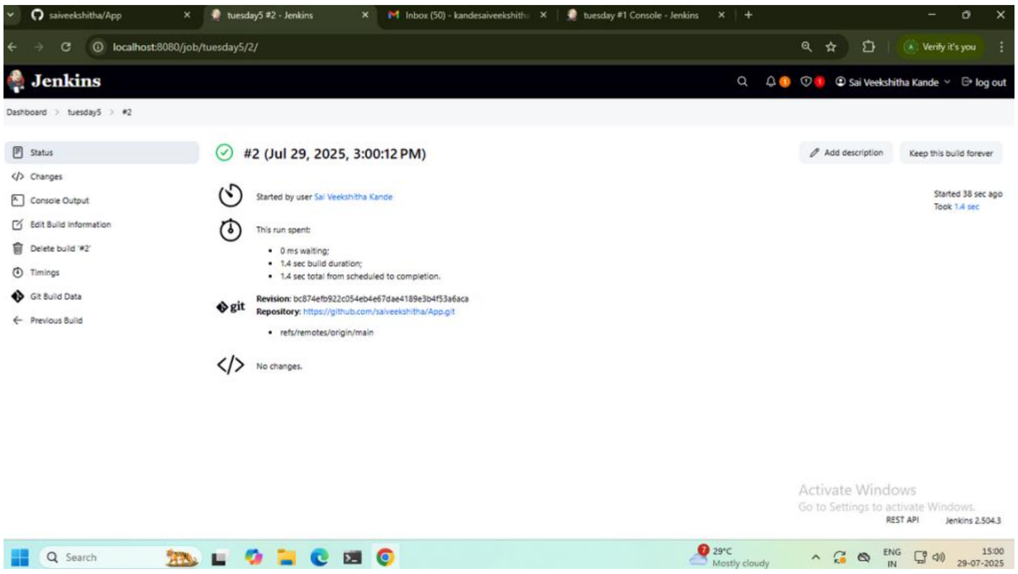
Permalinks

- Last build (#2), 3.8 sec ago
- Last stable build (#2), 3.8 sec ago
- Last successful build (#2), 3.8 sec ago
- Last completed build (#2), 3.8 sec ago

Builds

Today

- #2 3:00 PM
- #1 2:58 PM



localhost:8080/job/tuesday5/2/

Dashboard > tuesday5 > #2

#2 (Jul 29, 2025, 3:00:12 PM)

Started by user Sai Veekshitha Kande

Started 38 sec ago
Took 1.4 sec

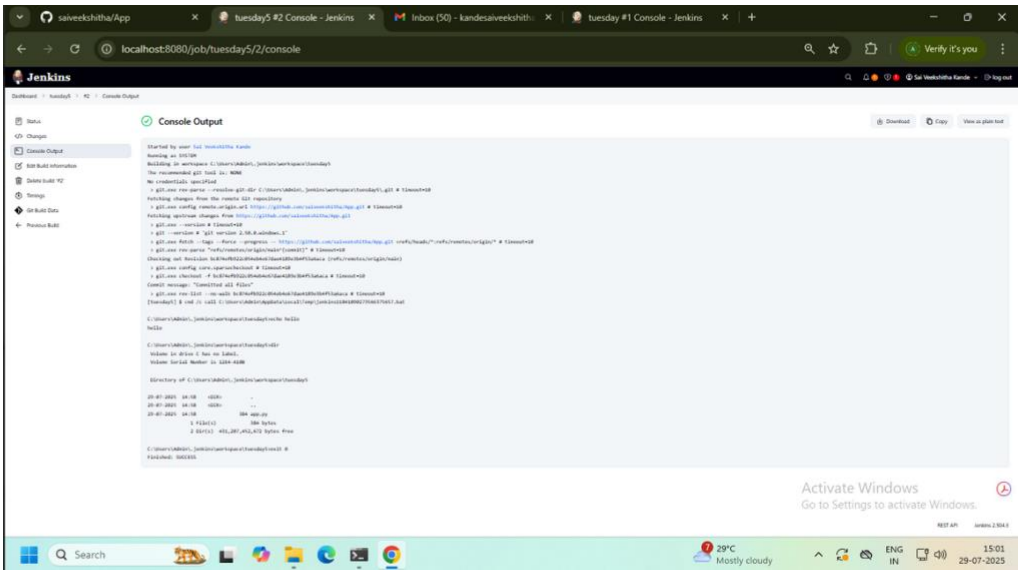
This run spent:

- 0 ms waiting;
- 1.4 sec build duration;
- 1.4 sec total from scheduled to completion.

git
Revision: bc874efb922c054eb4e67dae4189e3b4f53a6aca
Repository: https://github.com/saiveekshitha/App.git

- refs/remotes/origin/main

No changes.



localhost:8080/job/tuesday5/2/console

Console Output

```
Started by user Sai Veekshitha Kande
Running as SYSTEM
Building in workspace C:\Users\Admin\.jenkins\workspace\tuesday5
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\Users\Admin\.jenkins\workspace\tuesday5\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/saiveekshitha/App.git # timeout=10
Fetching upstream changes from https://github.com/saiveekshitha/App.git
 > git.exe --version # timeout=10
 > git --version # 'git version 2.50.0.windows.1'
 > git.exe fetch --tags --force --progress -- https://github.com/saiveekshitha/App.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/main^{commit}" # timeout=10
Checking out Revision bc874efb922c054eb4e67dae4189e3b4f53a6aca (refs/remotes/origin/main)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f bc874efb922c054eb4e67dae4189e3b4f53a6aca # timeout=10
Commit message: "Committed all files"
 > git.exe rev-list --no-walk bc874efb922c054eb4e67dae4189e3b4f53a6aca # timeout=10
[tuesday5] $ cmd /c call C:\Users\Admin\AppData\Local\Temp\jenkins[illegible].bat

C:\Users\Admin\.jenkins\workspace\tuesday5>echo hello
hello

C:\Users\Admin\.jenkins\workspace\tuesday5>dir
 Volume in drive C has no label.
 Volume Serial Number is 1214-4100

 Directory of C:\Users\Admin\.jenkins\workspace\tuesday5

29-07-2025  14:58    <DIR>          .
29-07-2025  14:58    <DIR>          ..
29-07-2025  14:58               304 app.py
               1 File(s)            304 bytes
               2 Dir(s)  431,207,452,672 bytes free

C:\Users\Admin\.jenkins\workspace\tuesday5>exit 0
Finished: SUCCESS
```