

# IISER RESEARCH PROJECT

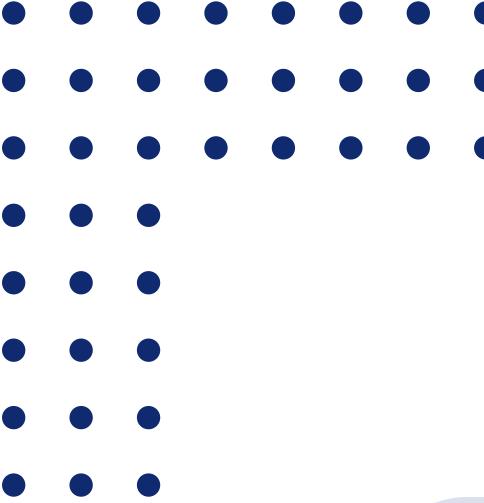


## DEVOPS PIPELINE

PRESENTED BY -

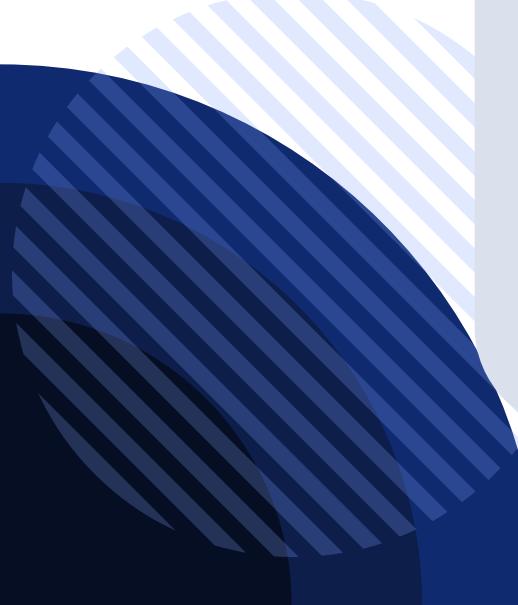
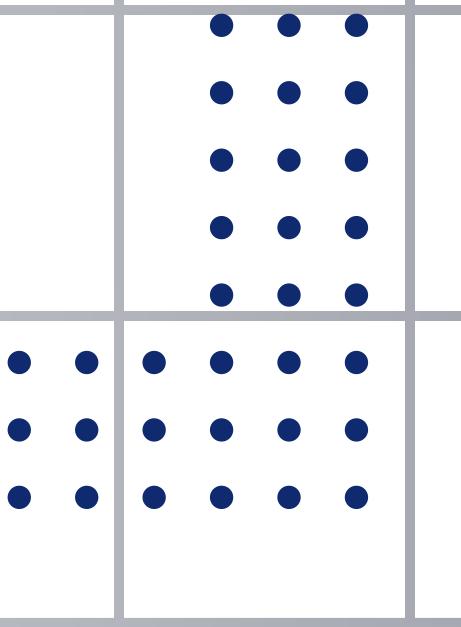
JOY BANERJEE	1023106
VIDUSHI BHARDWAJ	1023108
FAHEEM MADHIA	1023158
RANJEET MALI	1023160

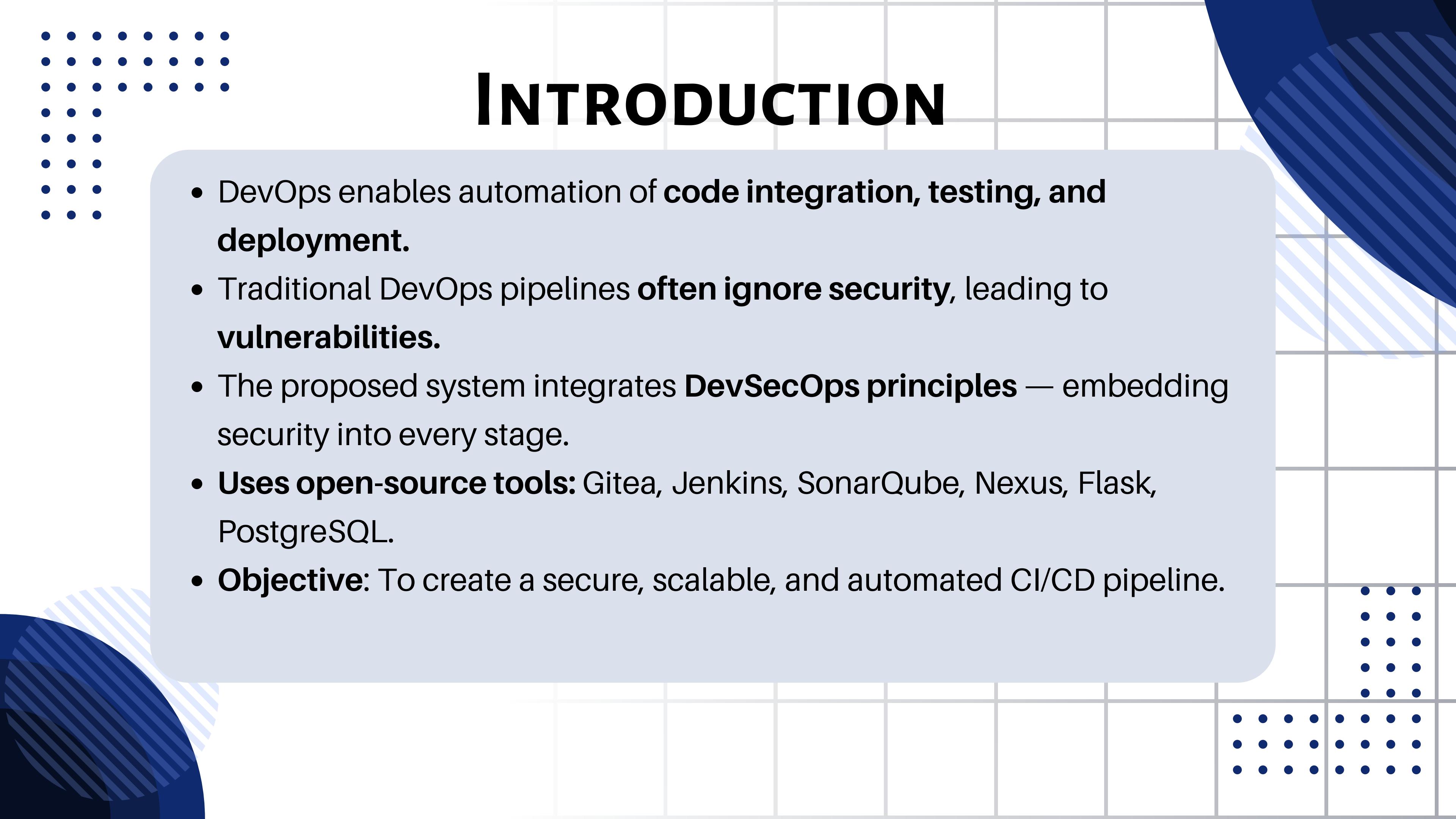
TE COMP GROUP 2  
2025 - 26



# LOCAL GOOGLE DRIVE-LIKE FILE STORAGE SYSTEM WITH DEVOPS PIPELINE



- Introduction
  - Motivation
  - Problem Statement
  - Methodology
  - System Architecture
  - Tools/Tech Stack
  - Literature Survey
  - Existing Survey
- 
- 



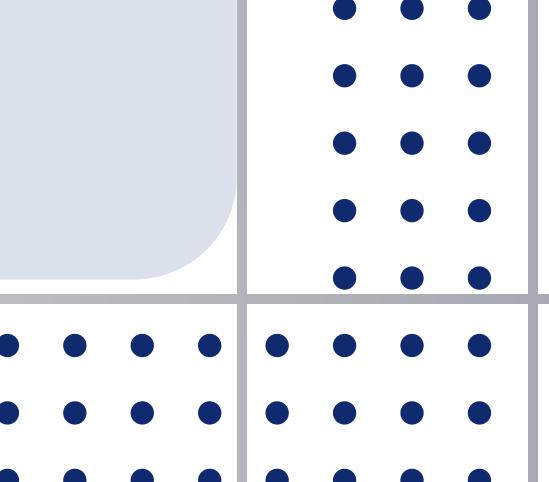
# INTRODUCTION

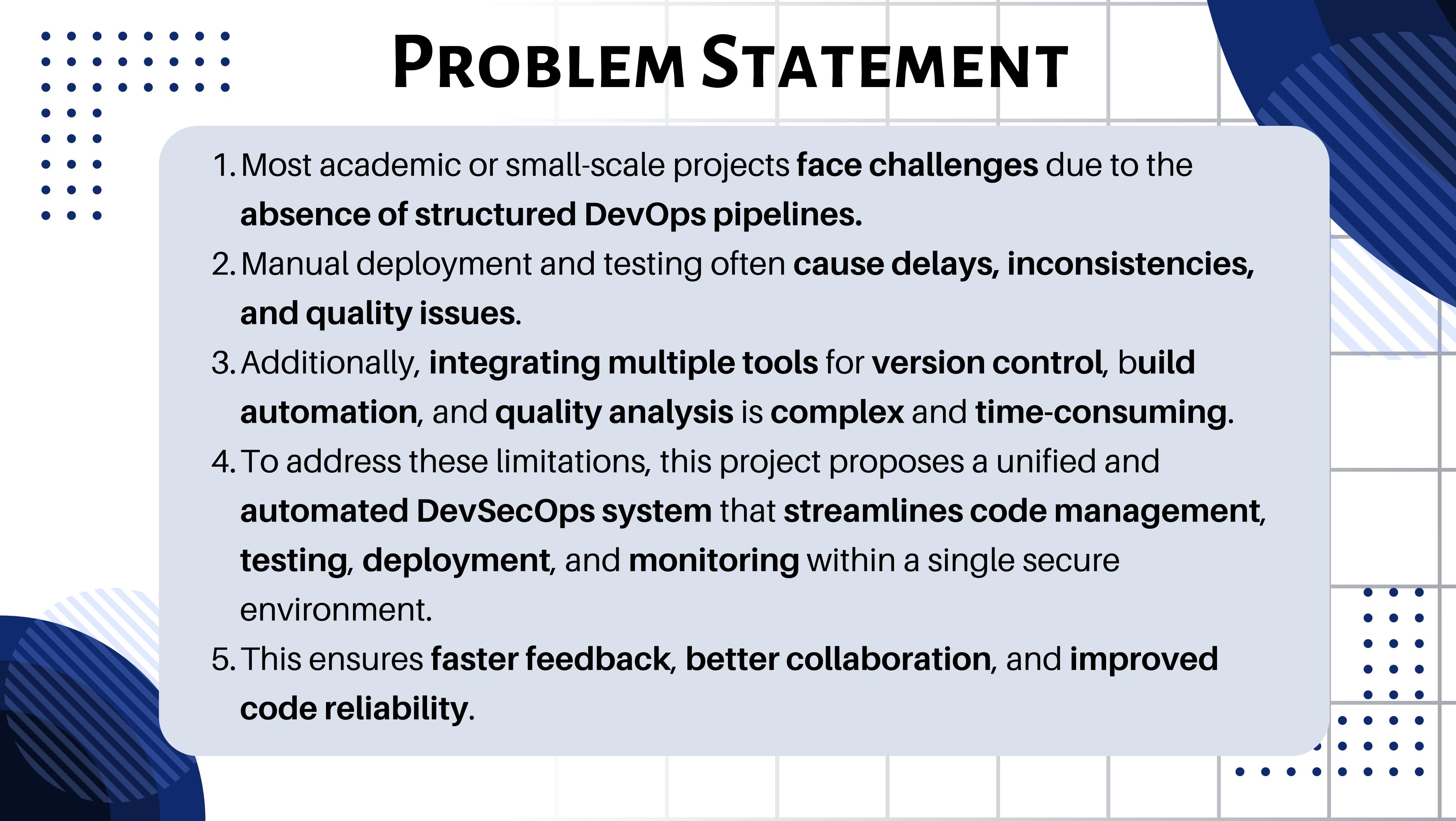
- DevOps enables automation of **code integration, testing, and deployment**.
- Traditional DevOps pipelines **often ignore security**, leading to **vulnerabilities**.
- The proposed system integrates **DevSecOps principles** — embedding security into every stage.
- **Uses open-source tools:** Gitea, Jenkins, SonarQube, Nexus, Flask, PostgreSQL.
- **Objective:** To create a secure, scalable, and automated CI/CD pipeline.



# MOTIVATION



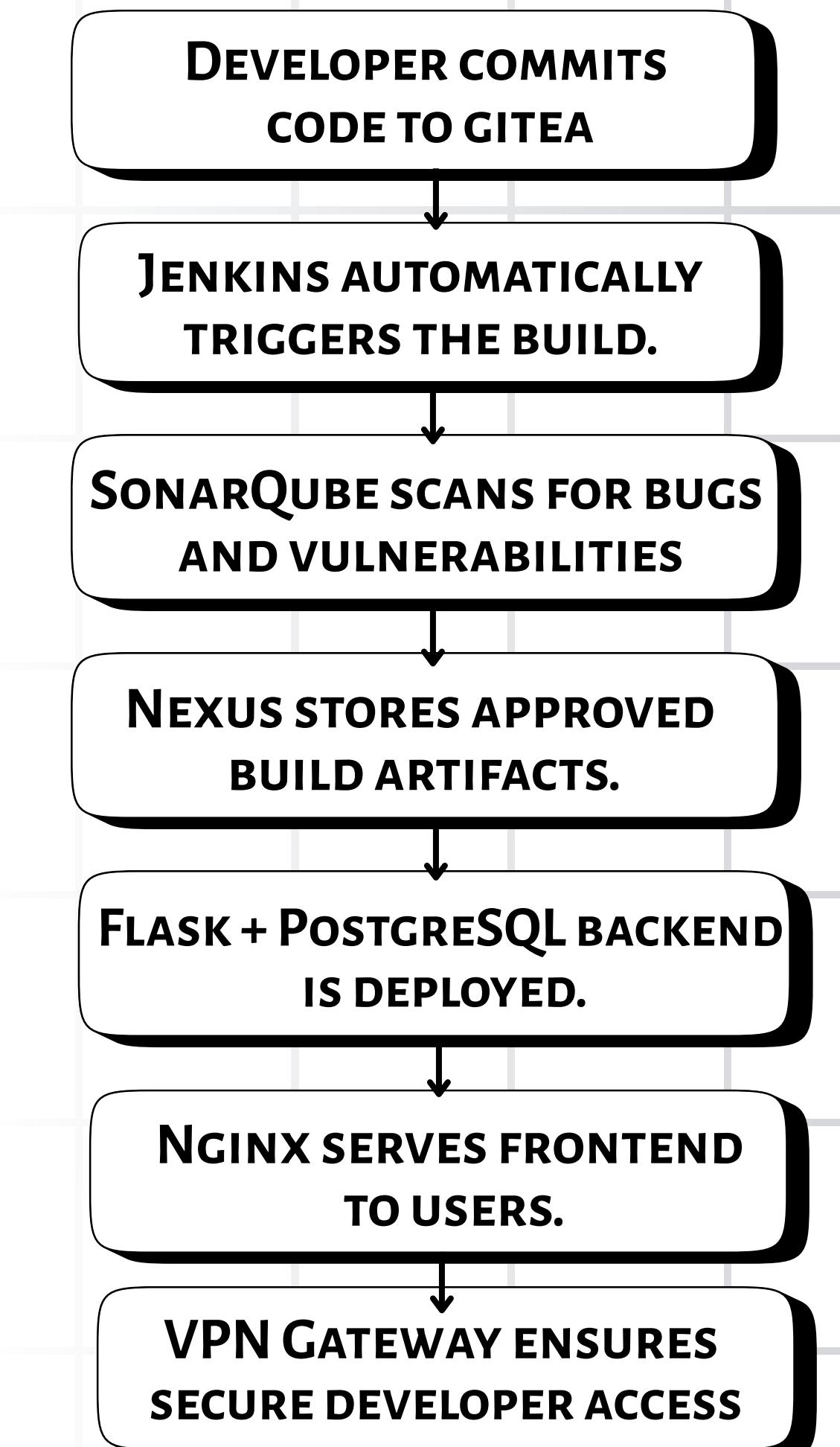
- Increasing cyber threats make secure automation essential.
  - Many existing academic projects rely on manual deployment.
  - Motivation: Integrate open-source tools into a unified DevSecOps pipeline.
  - Encourages a security-first mindset among developers.
  - Provides hands-on experience with real-world automation workflows.
- 
- 



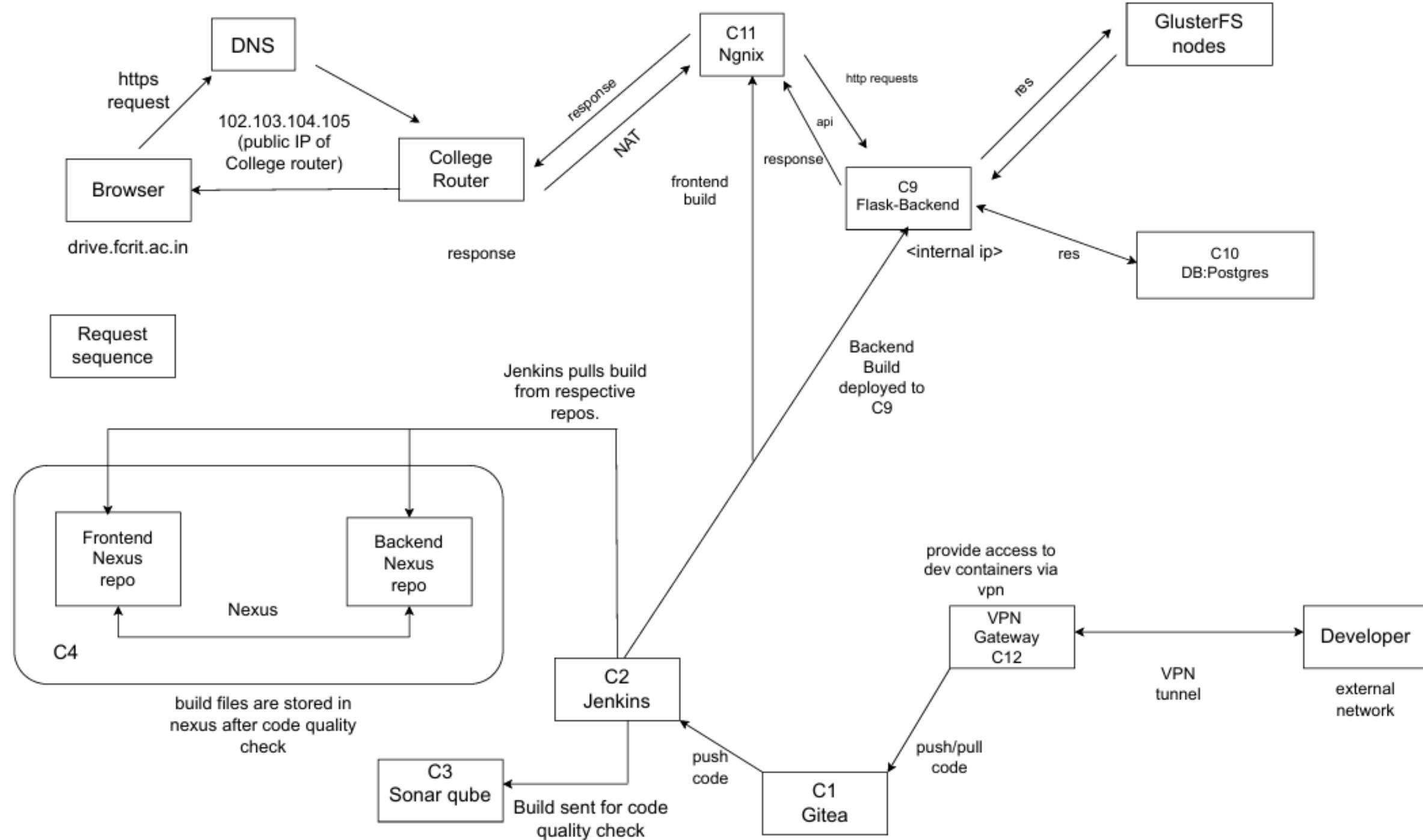
# PROBLEM STATEMENT

1. Most academic or small-scale projects **face challenges** due to the **absence of structured DevOps pipelines**.
2. Manual deployment and testing often **cause delays, inconsistencies, and quality issues**.
3. Additionally, **integrating multiple tools** for **version control, build automation, and quality analysis** is **complex and time-consuming**.
4. To address these limitations, this project proposes a unified and **automated DevSecOps system** that **streamlines code management, testing, deployment, and monitoring** within a single secure environment.
5. This ensures **faster feedback, better collaboration, and improved code reliability**.

# METHODOLOGY



# SYSTEM ARCHITECTURE



# Steps in Progress & steps to be done

PC 1		PC 2		PC 3		PC 4	
Container 1	✓	Container 2	✓	Container 1	✓	Container 2	
1. Jenkins:		1. Sonarqube:		1. GITEa:		1. Nexus:	
Working		Working		Working		Working	
Testing	✓	Testing	✓	Testing	✓	Testing	
<hr/>							
Container 1	✓	Container 2	✓	Container 1		Container 1	✓
1. Nginx:		Flask API		1. PostgreSQL		1. PostgreSQL	
Working		Working		Working		Working	
Testing	✓	Testing	✓	Testing		Testing	✓

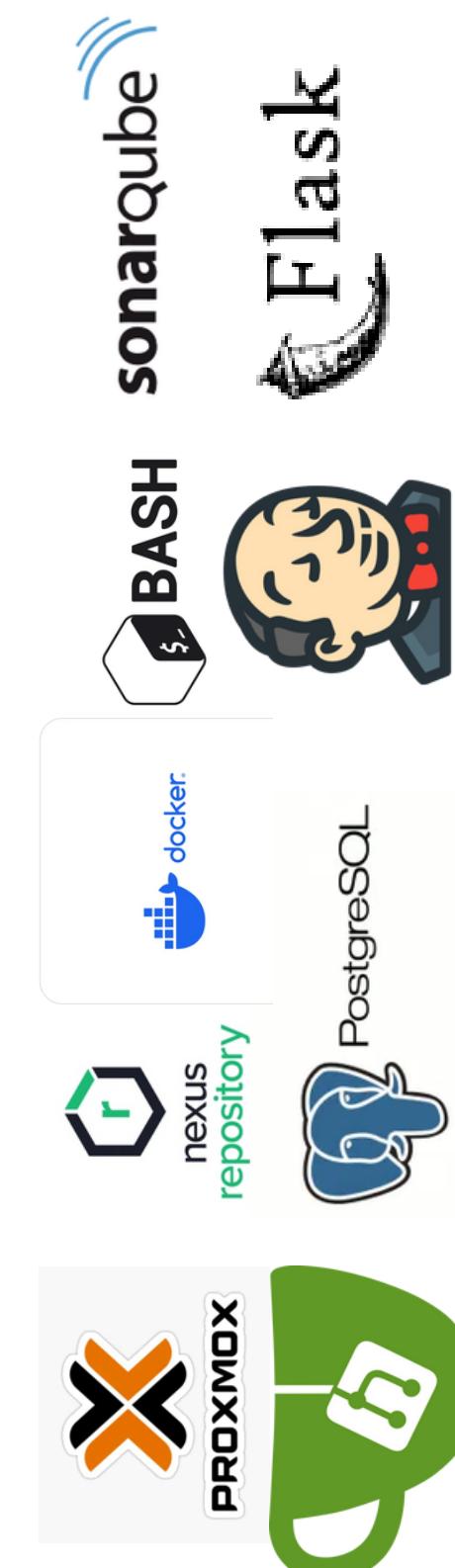
# TOOLS/TECHSTACK

## HARDWARE COMPONENTS

- 1. Server Nodes (4 PCs):**
  - Processor: Intel i5/i7 or Ryzen 5/7
  - RAM: 8-32 GB
  - Storage: 512 GB-1 TB SSD
  - Network: Gigabit Ethernet
  - Power: UPS backup
- 2. Client Systems:**
  - Processor: i3/i5
  - RAM: 4-8 GB
  - Stable LAN or Wi-Fi
- 3. Networking:**
  - Gigabit switch, Cat6 cables, Router

## SOFTWARE COMPONENTS

- 1. Operating System:** Ubuntu Server LTS
- 2. Version Control:** Gitea
- 3. Automation:** Jenkins
- 4. Code Analysis:** SonarQube
- 5. Artifact Repository:** Nexus
- 6. Backend:** Flask (Python)
- 7. Database:** PostgreSQL
- 8. Storage:** GlusterFS
- 9. Virtualization:** Proxmox VE
- 10. Additional Tools:** Docker, Git, SSH, Shell scripts



# STEPS IN PROGRESS

8/30/25, 4:56 PM

#25 - joy-portfolio-pipeline - Jenkins

joy-portfolio-pipeline #25 Pipeline Overview

Rerun

Manually run by Joy Banerjee Started 17 hr ago Queued 4 ms Took 50 min

Graph

```
graph LR; Start((Start)) --> SCM[Checkout SCM]; SCM --> Checkout[Checkout]; Checkout --> Install[Install All...]; Install --> SonarQube[SonarQube Analysis]; SonarQube --> QualityGate[Quality Gate Check]; QualityGate --> Reinstall[Reinstall Dependencies]; Reinstall --> Build[Build Project];
```

Search

- Checkout SCM 14s
- Cleaning up workspace... 0.22s
- Checkout 4.8s
- Delete workspace when build is done 3m 39s
  - [MS-CLEANUP] Deleting project workspace...
  - [MS-CLEANUP] Deferred wipeout is used...
  - [MS-CLEANUP] done
- Install All Dependencies 3m 40s
- SonarQube Analysis 34m
- Quality Gate Check 1.9s
- Reinstall Dependencies for Build 4m 53s
- Build Project 5.3s
- Deploy to Netlify 3m 11s
- Post Actions 3m 40s

joysource Projects Issues Roles Quality Profiles Quality Gates Administration

Search for projects... A

Jay Portfolio main

OVERVIEW Issues Security Hotspots Measures Code Activity

Project Settings Project Information

QUALITY GATE (STATUS: Passed) All conditions passed.

MEASURES

New Code	Overall Code	
1 Bug	Reliability	
0 Vulnerabilities	Security	
6 Security Hotspots	0.0% Reviewed	Security Review
52min Debt	6 Code Smells	Maintainability
0.0% Coverage on 261 Lines of code	0.0% Unit Tests	Duplicated Blocks

ACTIVITY

Choose graph type: Duplications Lines of Code Duplicated Lines

August 30, 2025 at 4:56 PM

Activity

# LITERATURE SURVEY

Title	Authors / Year	Key Focus	Relevance to Our Work
High Availability Cluster Design Using Proxmox VE	Sharma et al. (2021)	VM clustering and redundancy with Proxmox	Used to understand HA design with Proxmox
Virtualization in Education Infrastructure Using Open Source Tools	Ahmed & Patel (2020)	Proxmox and virtualization in academia	Validates feasibility of virtual infra in colleges
Containerization vs Virtualization Performance Comparison	Liu et al. (2022)	Benchmarking VMs vs containers	Supports choice of containers for modularity
KVM vs. LXC: Comparing Performance and Isolation of Hardware-assisted Virtual Routers	Rathore, Hidel & Sjödin (2013)	This study evaluates how LXC containers and KVM VMs perform, especially under hardware virtualization enhancements like SR-IOV and Intel VT-d. Although focused on virtual routers, its findings are broadly relevant to service hosting in virtualization environments	<b>LXC containers offer superior performance and reduced resource overhead.</b> Unless strict isolation or multi-OS support is needed, <b>KVM VMs don't offer significant advantage</b> for your current setup.

# EXISTING SYSTEMS

System	Description	Key Features	Limitations
<b>GitHub</b>	Cloud-based platform for version control and collaborative development using Git.	- Repository hosting with pull requests and issue tracking.- Integrates with CI/CD tools like Jenkins and GitHub Actions.- Supports open-source collaboration.	- Limited customization in free tier.- Requires internet access.- Advanced features available only in paid plans.
<b>GitLab</b>	Complete DevOps platform offering integrated CI/CD, monitoring, and code management.	- Built-in pipelines for automation.- Integration with Kubernetes and container registries.- Provides project and user access management.	- High resource usage when self-hosted.- Complex setup for enterprise use.- Some advanced options are paid.
<b>Travis CI</b>	Cloud-based CI/CD tool that automates code building and testing after every commit.	- Easy YAML-based configuration.- Supports multiple languages and frameworks.- Simple GitHub integration.	- Slow builds in free tier.- Limited environment control.- Less active community support.
<b>CircleCI</b>	Modern CI/CD platform supporting cloud and on-prem automation.	- Docker-based builds with parallel execution.- Fast and scalable pipelines.- Integration with GitHub and Bitbucket.	- Limited free-tier build minutes.- Complex for multi-container workflows.- Weak offline/on-premise support.

# CONCLUSIONS AND FUTURE SCOPE

## Conclusion:

- Achieving a secure, automated CI/CD pipeline using 3-tier architecture.
- Reduced manual work, improved quality and speed of deployment.
- Demonstrated scalable and fault-tolerant DevOps infrastructure.

&

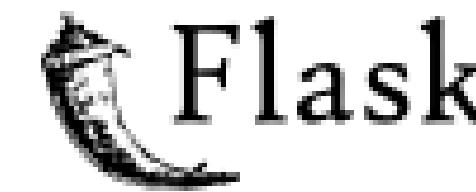
## Future Scope:

- Integration with Docker & Kubernetes.
- Monitoring using Prometheus + Grafana.
- Cloud deployment on AWS or GCP.
- Add role-based access and centralized logging.

# REFERENCE

- Andrew S. Tanenbaum and Maarten Van Steen. Distributed Systems: Principles and Paradigms. Pearson Education, 2016.
- Ian Sommerville. Software Engineering. Pearson Education Limited, 10th edition, 2016.
- Len Bass, Ingo Weber, and Liming Zhu. DevOps: A Software Architect's Perspective. Addison-Wesley Professional, 2015.
- K. Verma and V. Rajaraman. Cloud Computing: Concepts, Technology & Architecture. McGraw Hill Education, 2020.
- Ramez Elmasri and Shamkant B. Navathe. Fundamentals of Database Systems. Pearson Education, 7th edition, 2016.
- Docker Documentation. Available at: <https://docs.docker.com/>, 2025. Accessed: November 2025.
- Kubernetes Documentation. Available at: <https://kubernetes.io/docs/>, 2025. Accessed: November 2025.
- Amazon Web Services Documentation. Available at: <https://docs.aws.amazon.com/>, 2025. Accessed: November 2025.
- PostgreSQL Documentation. Available at: <https://www.postgresql.org/docs/>, 2025. Accessed: November 2025.
- Flask: Python Micro Web Framework Documentation. Available at: <https://flask.palletsprojects.com/>, 2025. Accessed: November 2025.

**THANK YOU**



Title	Authors / Year	Key Focus	Relevance to Our Work
High Availability Cluster Design Using Proxmox VE	Sharma et al. (2021)	VM clustering and redundancy with Proxmox	Used to understand HA design with Proxmox
Virtualization in Education Infrastructure Using Open Source Tools	Ahmed & Patel (2020)	Proxmox and virtualization in academia	Validates feasibility of virtual infra in colleges
Containerization vs Virtualization Performance Comparison	Liu et al. (2022)	Benchmarking VMs vs containers	Supports choice of containers for modularity
KVM vs. LXC: Comparing Performance and Isolation of Hardware-assisted Virtual Routers	Rathore, Hidel & Sjödin (2013)	This study evaluates how LXC containers and KVM VMs perform, especially under hardware virtualization enhancements like SR-IOV and Intel VT-d. Although focused on virtual routers, its findings are broadly relevant to service hosting in virtualization environments	<b>LXC containers offer superior performance and reduced resource overhead.</b> Unless strict isolation or multi-OS support is needed, <b>KVM VMs don't offer significant advantage</b> for your current setup.