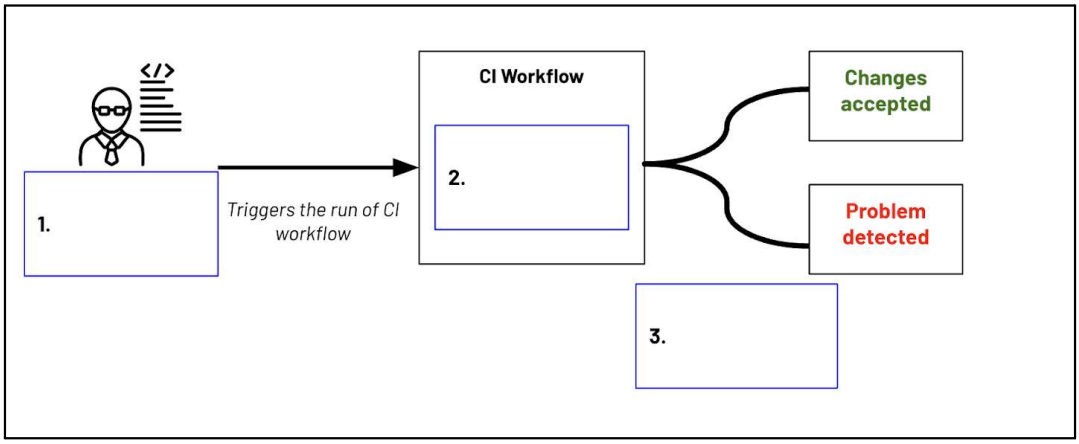
MLA// EXTENDED ASSESSMENT TASK

Please make a copy of this document

NAME OF SESSION	DEVELOPMENT: CI & CODE QUALITY
NAME OF STUDENT	Funkeyi Jessica Omoro

ASSESSMENT FOCUS	Continuous Integration (CI) Pipeline
TASK	Assume you have a project that has the following features: - Three microservices (X, Y and Z) - Three environments (Dev, Test, Pre-Prod) Create a CI pipeline to be triggered from a commit to the main branch in *GIT*, and end with a binary that is ready and can be deployed to the production environment. Draw the CI pipeline and discuss, in detail, all the logic for the different processes involved.

1. **Use the diagram below to outline your CI pipeline. For each of the blue boxes, provide a description of the step, along with an explanation of the logic involved.**



1 = Source Control & Trigger

Description: pipeline initiation and source code preparation

- Trigger Event: Developer commits/merges to main branch
- Logic:
    1. Git webhook triggers CI system(Jenkins, GitLab CI, Github Actions)

MLA// EXTENDED ASSESSMENT TASK

Please make a copy of this document

2. Pipeline fetches latest code from main branch
3. Determines which microservices changed using path-based triggers or dependency analysis
4. Creates unique build identifier (commit SHA + timestamp)

2 = Pre-Build Validation

Description: Fast feedback loop for basic code quality

Logic:

1. Linting: check code style and formatting for all three services
2. **Static analysis:** run static scans, code complexity and analysis
3. **Dependency check:** verify all dependencies are available and secure
4. **Parallel execution:** run checks for X, Y, Z simultaneously to save time
5. **Fast fail:** stop pipeline immediately if critical issues found

3 = Build and package

Description: compile and containerise each microservice

Logic:

- **Compile Code:** Build each microservice (X, Y, Z) using appropriate build tools (Maven, npm, etc.)
- **Create Docker Images:** Package each service into lightweight, production-ready containers
- **Tagging Strategy:** Tag images with version number and commit SHA for traceability
- **Parallel Processing:** Build all three services simultaneously to reduce total build time
- **Artifact Storage:** Push built images to container registry for later deployment stages

4 = Testing

Description: Automated testing validation

Logic:

- **Unit Tests:** Execute isolated tests for each microservice's business logic
- Integration Tests: Test inter-service communication between X, Y, and Z
- Contract Testing: Verify API contracts and service dependencies
- Test Isolation: Use test databases and mock external services
- Coverage Validation: Ensure minimum test coverage thresholds are met

MLA// EXTENDED ASSESSMENT TASK

Please make a copy of this document

5 = Deploy to Dev

Description: Deploy to development environment for initial validation

Logic:

- **Environment Setup:** Deploy all three microservices to Dev environment
- Service Dependencies: Ensure proper startup order and service discovery
- Health Checks: Verify all services are running and can communicate
- Smoke Tests: Run basic functionality tests to validate deployment
- Database Setup: Apply any schema changes or seed data

6 = Deploy to Test

Description: Deploy to test environment for comprehensive testing

Logic:

- **Environment Promotion:** Deploy same artifacts validated in Dev
- End-to-End Testing: Execute full user journey and system integration tests
- Performance Testing: Run load tests and validate response times
- Regression Testing: Ensure new changes don't break existing functionality
- Test Data Management: Use production-like test data for realistic validation

7 = Deploy to Pre-Prod

Description: Final validation in production-like environment

Logic:

- **Production Mirror:** Deploy to environment identical to production setup
- **Acceptance Testing:** Execute business acceptance criteria and user acceptance tests
- **Security Validation:** Run security scans and penetration testing
- **Performance Benchmarking:** Validate system meets SLA requirements
- **Monitoring Verification:** Ensure all logging and monitoring systems function correctly

8 = Artifact Ready for Production

Description: Create production-ready deployment package

Logic:

- **Artifact Publishing:** Push validated container images to production registry
- **Release Package Creation:** Bundle deployment configurations, scripts, and documentation
- **Version Tagging:** Apply final production version tags to all artifacts
- **Approval Gateway:** Require manual sign-off before marking as production-ready

MLA// EXTENDED ASSESSMENT TASK

Please make a copy of this document

- **Rollback Preparation:** Ensure previous version artifacts remain available for quick rollback