# DevOps Developer Flow: IntelliJ, GitHub, Jenkins, and Red Hat ACS

This document provides an in-depth overview of the DevOps development flow for Java developers. It covers the integration between IntelliJ IDEA, GitHub, Jenkins, and Red Hat Advanced Cluster Security (RHACS). The goal is to establish a secure, automated, and efficient CI/CD pipeline that ensures smooth deployments while enforcing security best practices.

## Table of Contents

1. Introduction  
2. Development in IntelliJ  
3. Version Control with GitHub  
4. Continuous Integration with Jenkins  
5. Container Image Building & Storage  
6. Security Scanning with RHACS  
7. Deployment to Test/Staging Environment  
8. Code Review and Pull Requests  
9. Production Deployment and Final Security Checks  
10. Ongoing Monitoring & Incident Response  
11. Best Practices for Secure DevOps  
12. Conclusion

## Development in IntelliJ

Developers use \*\*IntelliJ IDEA\*\* as the primary IDE for Java development, which offers:

* - Built-in Git integration for seamless version control.
* - Support for Maven and Gradle for dependency management.
* - Automated testing tools to run unit and integration tests before committing changes.
* - Code analysis tools for improving code quality and reducing technical debt.

## Version Control with GitHub

Developers use \*\*GitHub\*\* to manage code versions and collaborate effectively. Key steps:

* - Developers \*\*commit and push code\*\* from IntelliJ IDEA.
* - Feature branches are created for \*\*isolated development\*\*.
* - Pull Requests (PRs) are used for \*\*code review and validation\*\*.
* - GitHub Actions can be configured to \*\*run tests before merging PRs\*\*.

## Continuous Integration with Jenkins

Jenkins is configured to \*\*automate the build process\*\* upon code changes. Steps:

* - Jenkins \*\*monitors the GitHub repository\*\* for new commits.
* - It \*\*pulls the latest code\*\* and \*\*compiles Java\*\* using Maven/Gradle.
* - Unit tests are executed automatically to \*\*catch failures early\*\*.
* - Test results are reported, and if successful, the pipeline proceeds.

## Container Image Building & Storage

Once Jenkins validates the build, a \*\*Docker container image\*\* is created.

* - Jenkins runs a \*\*Dockerfile\*\* to package the Java application.
* - The container image is \*\*tagged with a version number\*\*.
* - The image is \*\*pushed to a container registry\*\* (e.g., DockerHub, Quay).

## Security Scanning with RHACS

RHACS performs security scanning \*\*before deployment\*\* to detect vulnerabilities.

* - Jenkins triggers an \*\*RHACS scan\*\* after building the container image.
* - RHACS checks for \*\*known vulnerabilities (CVEs)\*\*.
* - Policy violations (e.g., \*\*running as root, outdated libraries\*\*) are flagged.
* - Developers receive a \*\*report of security risks\*\*, blocking deployment if needed.

## Deployment to Test/Staging Environment

If RHACS scans pass, the application is deployed to \*\*a test environment\*\* for further validation.

* - Jenkins deploys the \*\*containerized application\*\* to a test Kubernetes cluster.
* - QA engineers run \*\*integration and performance tests\*\*.
* - RHACS continues monitoring \*\*running containers for security threats\*\*.

## Code Review and Pull Requests

Before merging changes into production, a \*\*pull request (PR) review process\*\* is followed.

* - Developers submit \*\*pull requests in GitHub\*\*.
* - Team members \*\*review code for quality and best practices\*\*.
* - Jenkins runs \*\*build and tests on the PR branch\*\*.
* - Approval from reviewers allows merging the PR into the main branch.

## Production Deployment and Final Security Checks

Once changes pass testing, Jenkins \*\*deploys to production\*\*.

* - A final \*\*RHACS scan ensures compliance\*\*.
* - Jenkins builds the final \*\*production-ready container\*\*.
* - If security checks pass, the image is \*\*deployed to Kubernetes in production\*\*.

## Ongoing Monitoring & Incident Response

RHACS continues to monitor production containers for security threats.

* - Security alerts notify teams of \*\*new vulnerabilities\*\*.
* - RHACS enforces \*\*policy compliance\*\* in real-time.
* - Incident response is triggered for \*\*critical security threats\*\*.

## Conclusion

This document outlines an automated, secure DevOps workflow for Java developers using IntelliJ, GitHub, Jenkins, and Red Hat ACS. Following best practices in CI/CD and security scanning ensures efficient, reliable, and secure deployments.