To automate **SAST (Static Application Security Testing)** and its reporting efficiently across one or multiple applications, the key is to build a modular, reusable DevSecOps pipeline integrated with your CI/CD process (like Jenkins). Below is a **clear roadmap** with best practices and tools:

**✅ Roadmap for Automating SAST Reporting**

**🧭 Phase 1: Setup for a Single Application**

**1. Define Goals**

* Automatically scan code for security issues (like SQLi, XSS, etc.)
* Automatically remediate (if possible)
* Generate human-readable reports
* Open Pull Requests (PRs) with suggested fixes
* Notify security/review teams

**2. Choose SAST Tools**

You can combine open-source and commercial tools based on your budget.

| **Purpose** | **Tool** | **Notes** |
| --- | --- | --- |
| SAST scanning | **Semgrep** | Fast, open-source, customizable rules |
| SAST scanning | **SonarQube** | Code quality + security; good GUI |
| Dependency Scan | **OWASP Dependency-Check** | For vulnerable 3rd-party libs |
| SAST + Remediation | **Snyk Code** (optional) | Commercial, with PRs for fixes |

**3. Integrate into CI/CD (e.g., Jenkins)**

Your Jenkinsfile should include:

* Checkout code
* Run SAST tools
* Auto-fix issues (if tool supports it)
* Generate report (HTML, SARIF, JSON, etc.)
* Commit fixes to new branch
* Create PR with fix & report
* Notify reviewer

Example Jenkins Steps (for Semgrep):

bat 'semgrep scan --config .semgrep/sql-injection-autofix.yml --autofix --json > semgrep\_report.json'

bat 'gh pr create --title "Fix: SAST Auto-remediation" --body "Auto-fixed security issues.\nPlease review the attached report." --base main'

**4. Generate and Store Reports**

* Generate HTML/JSON reports using tools like:
  + semgrep --json
  + dependency-check --format HTML
* Archive them in Jenkins:

archiveArtifacts artifacts: '\*\*/semgrep\_report.json', allowEmptyArchive: true

* Optional: Push to external dashboard (ELK, Splunk, S3, or email)

**✅ Phase 2: Make It Reusable for Multiple Apps**

**5. Build a Jenkins Shared Library**

Move logic from Jenkinsfile to reusable shared library:

jenkins-shared-lib/

├── vars/

│ └── cicdPipeline.groovy # Main entry point

├── src/org/example/

│ ├── SemgrepHandler.groovy # Run semgrep, parse output

│ ├── GitHandler.groovy # Git checkout, branch, PR

│ ├── ReportHandler.groovy # Report parsing & upload

Then in each app’s Jenkinsfile:

@Library('jenkins-shared-lib') \_

cicdPipeline()

**6. Parameterize the Pipeline**

Let each app provide:

* App name
* Semgrep config path
* GitHub repo URL
* Deployment toggle
* Teams to notify

Use a YAML or JSON file per repo:

appName: employee-service

semgrepConfig: .semgrep/sql-injection-autofix.yml

repoUrl: https://github.com/org/employee-service

notifyTeam: security-team

**7. Apply for Other Applications**

Once your framework is tested on one app:

1. Add new app config
2. Reuse shared library
3. Onboard app into Jenkins
4. Review reports and PRs

**🛠 Recommended Tools Stack**

| **Category** | **Tools** |
| --- | --- |
| **CI/CD** | Jenkins, GitHub Actions |
| **SAST** | Semgrep, SonarQube, Snyk |
| **Dependency Scan** | OWASP Dependency-Check, Snyk |
| **PR Automation** | GitHub CLI (gh), GitLab API |
| **Reporting** | Jenkins artifacts, HTML reports, Slack/email notification |
| **Security Workflow** | Jira integration (for triaging), Approval steps in PRs |

**📊 Optional Add-ons**

* **Slack or MS Teams notifications** for PRs or vulnerabilities
* **Jira ticketing** for critical issues
* **S3/ElasticSearch** centralized report archiving
* **GitHub Code Scanning Alerts** with SARIF upload

**✅ Final Thoughts**

| **Step** | **Action** |
| --- | --- |
| 1 | Try Semgrep + auto-fix on one app |
| 2 | Create Jenkins shared library |
| 3 | Add GitHub PR + HTML/JSON report |
| 4 | Make the process configurable |
| 5 | Onboard other apps using config files |
| 6 | Notify security team for manual review |
| 7 | Continuously improve rules and coverage |

Would you like a starter template for the Jenkins shared library or SemgrepHandler.groovy?