[COMPANY LOGO]

Application Intelligence Report

Comprehensive Analysis and Migration Assessment

Repository: https://github.com/end-of-game/openshift-voting-app

Analysis Date: July 18, 2025

*Generated by Application Intelligence Platform*

Table of Contents

Executive Summary........................................... 3

Application Overview........................................ 4

Component Analysis.......................................... 5

Architecture Assessment..................................... 8

Security Analysis........................................... 10

Git History Analysis........................................ 12

Recommendations............................................. 13

Appendices.................................................. 15

Executive Summary

|  |  |
| --- | --- |
| **Metric** | **Value** |
| Total Components | 3 |
| Programming Languages | python, java, nodejs |
| Containerization Status | 3 containerized |
| Data Sources | 0 |
| Security Findings | 0 |
| Git Commits | 1 |
| Architecture Style | microservices |

Application Overview

This report presents a comprehensive analysis of the application repository. The analysis identified 3 components using 3 different programming languages. The application demonstrates a microservices architecture pattern.

Key Findings

• 📦 3 application components identified

• 🔧 3 programming languages detected: python, java, nodejs

• 🐳 3 components are containerized

• 💾 0 data sources identified

• 🔒 0 security findings require attention

Detailed Analysis

Component Analysis

The analysis identified 3 components across the application:

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | **Language** | **Type** | **Packaging** |
| vote | python | Unknown | docker |
| worker | java | Unknown | docker |
| result | nodejs | Unknown | docker |

Component: vote

• Language: python

• Runtime: python

• Build Tool: unknown

• Packaging: docker

• Exposed Ports: 8080

• Base Images: python:3.9-slim

Component: worker

• Language: java

• Runtime: java

• Build Tool: unknown

• Packaging: docker

• Base Images: openjdk:8-jre, maven:3.5-jdk-8-alpine

**Notes:**

• Alternative C# implementation found at 'worker/src/src/Worker/Program.cs' but does not appear to be the primary build target defined in the Dockerfile or OpenShift manifests. Primary implementation is Java.

• Multiple base images detected: openjdk:8-jre, maven:3.5-jdk-8-alpine. This may indicate multi-stage builds or alternative build strategies.

Component: result

• Language: nodejs

• Runtime: nodejs

• Build Tool: unknown

• Packaging: docker

• Exposed Ports: 8080

• Base Images: node:10-slim

Architecture Analysis

Architecture Style: microservices (Confidence: ConfidenceLevel.HIGH)

Reasoning: Multiple components with independent deployment characteristics

**Evidence:**

• Found 3 components

• Multiple deployable components detected

• 3 containerized components

• Multiple deployment configurations

Security Analysis

Security analysis identified 2 findings with 3 base image risks.

**Key Security Findings:**

• Unknown: The 'result' component uses the node:10-slim base image, which is End-of-Life (EOL) and contains numerous unpatched vulnerabilities. This is a significant security risk as it is no longer supported and likely contains known exploits. (Severity: CRITICAL)

• Unknown: A potential hardcoded secret (e.g., password) was detected in the 'voting-app' source code. Hardcoding credentials poses a significant risk, as they can be exposed if the source code is compromised. (Severity: HIGH)

Git History Analysis

• Total Commits: 1

• Active Contributors: 0

• Recent Activity: inactive

• Code Stability: high

Recommendations

🔴 High Priority Recommendations

• 🔒 Security: 2 critical/high severity vulnerabilities found. Prioritize security remediation.

• 💼 Business Criticality: Assessment is inferred from technical indicators. Validate with business stakeholders.

🟢 Low Priority Recommendations

• 📊 Development Activity: Low recent activity detected. Consider reviewing development processes and team capacity.

• 🐳 Base Images: 3 base images have known risks. Update to more recent versions.

Appendices

Appendix A: Technical Details

This analysis was generated using the Application Intelligence Platform, which performs comprehensive analysis of application repositories including code structure, infrastructure configuration, and security assessment.

Appendix B: Analysis Methodology

• Component Discovery: Automated scanning of source code and configuration files

• Language Detection: Analysis of file extensions, build configurations, and base images

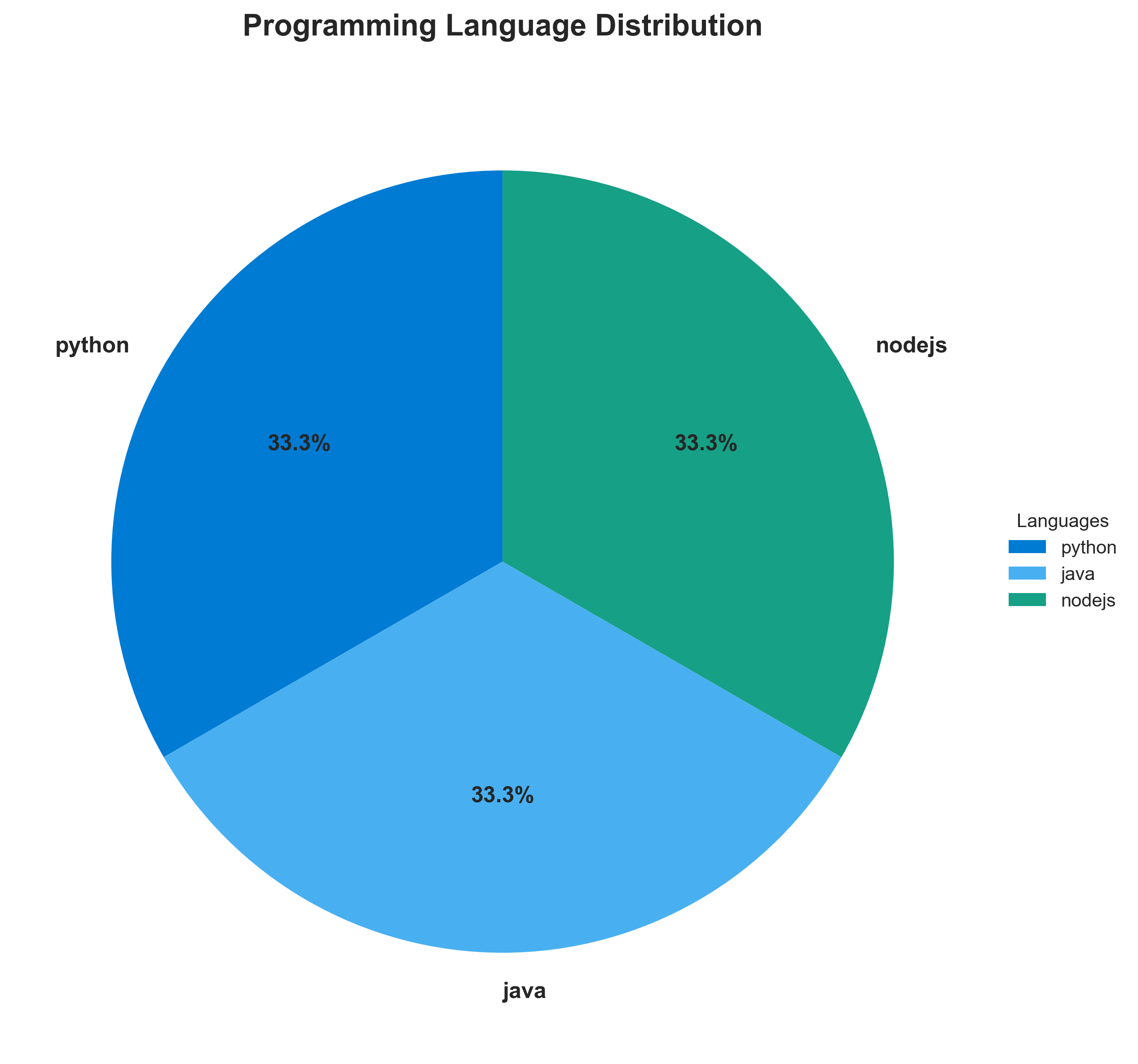
• Architecture Assessment: Evaluation of deployment patterns and component relationships

• Security Analysis: Scanning for common vulnerabilities and configuration issues

• Git History Analysis: Examination of commit patterns and development activity

Charts and Visualizations

Programming Language Distribution



**📊 Context:** This diagram illustrates the programming language distribution across the application's components, specifically identifying Python, Java, and Node.js. This analysis is crucial for our application intelligence report, informing modernization strategies and potential migration efforts.

**📊 Key Insights:** The application is a polyglot microservices architecture, utilizing three distinct programming languages: Python for the 'vote' component, Java for the 'worker' component, and Node.js for the 'result' component. All identified components are containerized using Docker, indicating a degree of modern deployment practices. Notably, the 'worker' component (Java) and the 'result' component (Node.js) are utilizing base images flagged as vulnerable ('openjdk:8-jre', 'maven:3.5-jdk-8-alpine', and 'node:10-slim' respectively), presenting immediate security risks.

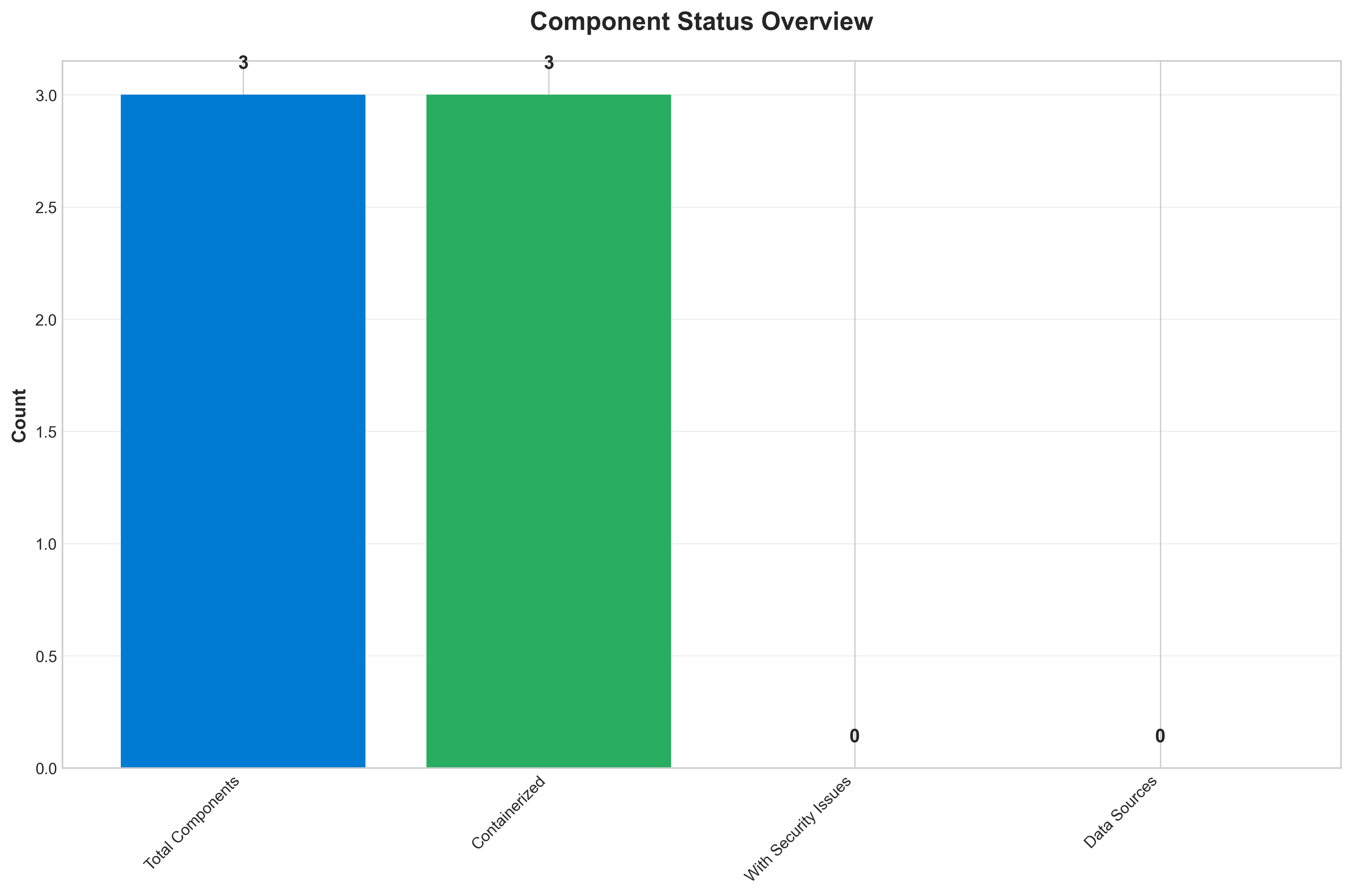
**📊 Business Impact:** The use of multiple languages presents both opportunities for leveraging specialized tooling and challenges for skill set consolidation during modernization. The identified vulnerabilities in base images for Java and Node.js components pose a significant security risk, potentially leading to data breaches or service disruptions. This necessitates urgent attention to mitigate these risks, which could impact project timelines and require additional investment in security patching or base image upgrades.

**📊 Recommendations:** Prioritize the immediate remediation of vulnerable base images for the 'worker' (Java) and 'result' (Node.js) components. This may involve upgrading to supported and secure versions or rebuilding the containers. Evaluate the long-term strategy for each language, considering whether to standardize on a single language or embrace the polyglot nature, which may influence the complexity and cost of future maintenance and development.

**📊 Technical Details:** The 'vote' component uses Python with a 'python:3.9-slim' base image. The 'worker' component is Java, using 'openjdk:8-jre' and 'maven:3.5-jdk-8-alpine' as base images, with a note about a potential C# implementation. The 'result' component is Node.js, relying on a 'node:10-slim' base image. All components are containerized (packaging: 'docker'), and the 'worker' component has a specific environment variable 'JAVA\_APP\_JAR' pointing to its executable.

────────────────────────────────────────────────────────────────────────────────

Component Status Overview



**📊 Context:** This Component Status Overview diagram provides an aggregated view of the application's constituent parts, specifically focusing on containerization and security posture. It was generated as part of a broader application intelligence report to inform modernization and migration planning efforts.

**📊 Key Insights:** All 3 identified application components are containerized. However, a critical security concern exists across all components, with 3 HIGH severity findings related to outdated and vulnerable base images. Specifically, the 'worker' component uses 'openjdk:8-jre' and 'maven:3.5-jdk-8-alpine', both identified with known vulnerabilities. The 'result' component is running on 'node:10-slim', which is End-of-Life and poses significant security risks. There are no medium or low severity findings, indicating a concentrated critical vulnerability issue.

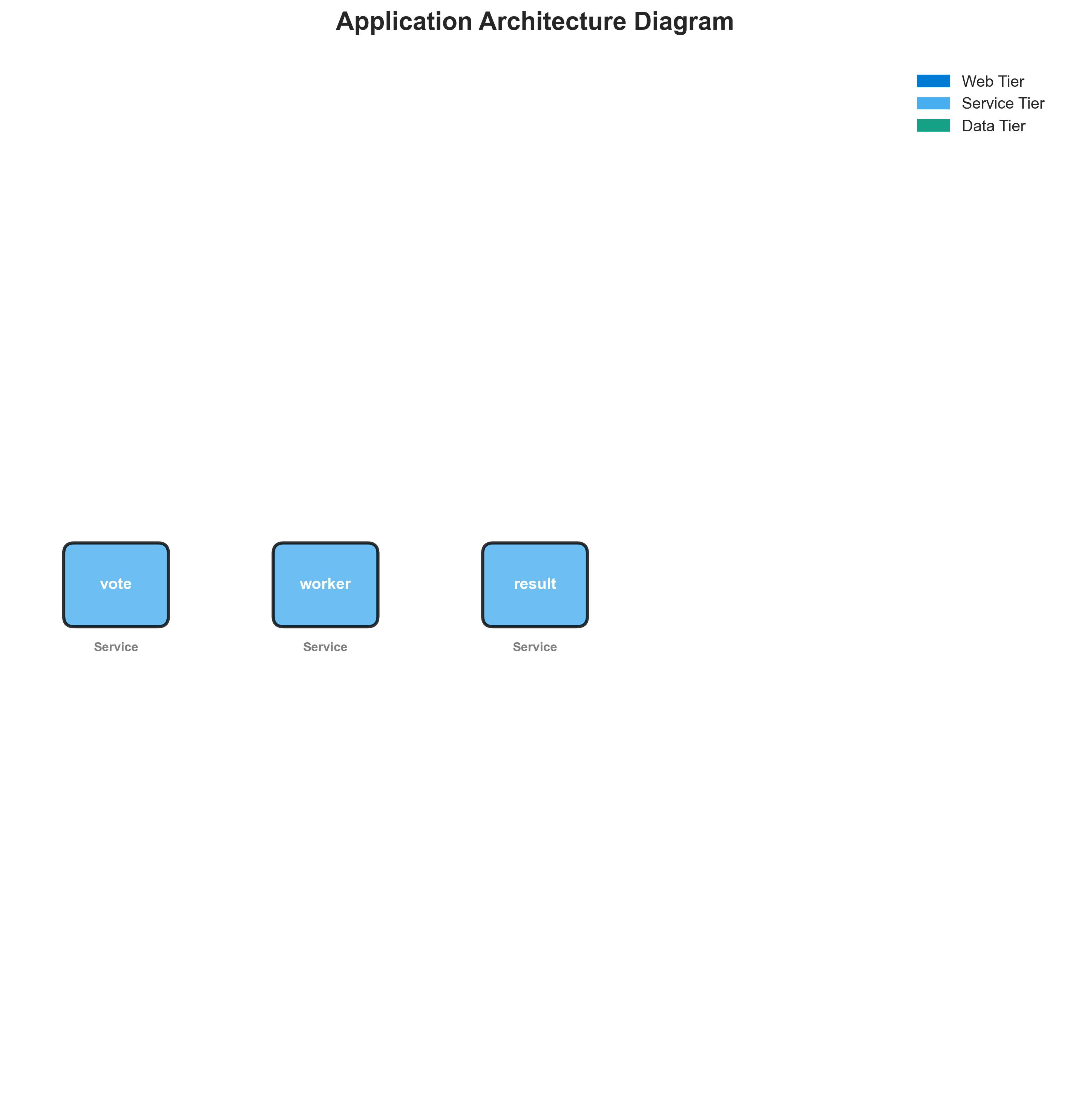
**📊 Business Impact:** The presence of HIGH severity vulnerabilities in all components significantly elevates the security risk profile of the application, making it a prime target for cyberattacks and potentially leading to data breaches or service disruptions. The use of EOL software ('node:10-slim') also creates compliance and support challenges, hindering future modernization efforts and potentially increasing operational costs. Addressing these vulnerabilities is a critical prerequisite for any successful migration or modernization strategy.

**📊 Recommendations:** Prioritize immediate remediation of all HIGH severity security findings by updating all base images to supported and patched versions. This includes updating 'openjdk:8-jre', 'maven:3.5-jdk-8-alpine', and 'node:10-slim'. Simultaneously, plan for a comprehensive vulnerability management program to proactively identify and address security issues in the future, ensuring ongoing compliance and security posture.

**📊 Technical Details:** The analysis highlights that all 3 components are containerized, simplifying potential cloud migration paths. The security findings are all rooted in the selection of base container images. The 'worker' component's dependencies on outdated Java and Maven versions, and the 'result' component's reliance on an EOL Node.js version, are the direct causes of the identified HIGH severity vulnerabilities. The 'base\_image\_analysis' scan method confirms the origin of these security concerns.

────────────────────────────────────────────────────────────────────────────────

Application Architecture



**📊 Context:** Architecture diagram shows {'value': 'microservices', 'confidence': <ConfidenceLevel.HIGH: 'HIGH'>, 'evidence': ['Found 3 components', 'Multiple deployable components detected', '3 containerized components', 'Multiple deployment configurations'], 'reasoning': 'Multiple components with independent deployment characteristics', 'limitations': ['Cannot determine communication patterns without runtime analysis']} pattern with 3 components.

**📊 Key Insights:** System follows {'value': 'microservices', 'confidence': <ConfidenceLevel.HIGH: 'HIGH'>, 'evidence': ['Found 3 components', 'Multiple deployable components detected', '3 containerized components', 'Multiple deployment configurations'], 'reasoning': 'Multiple components with independent deployment characteristics', 'limitations': ['Cannot determine communication patterns without runtime analysis']} architecture with unknown operational complexity. Offers requires further analysis.

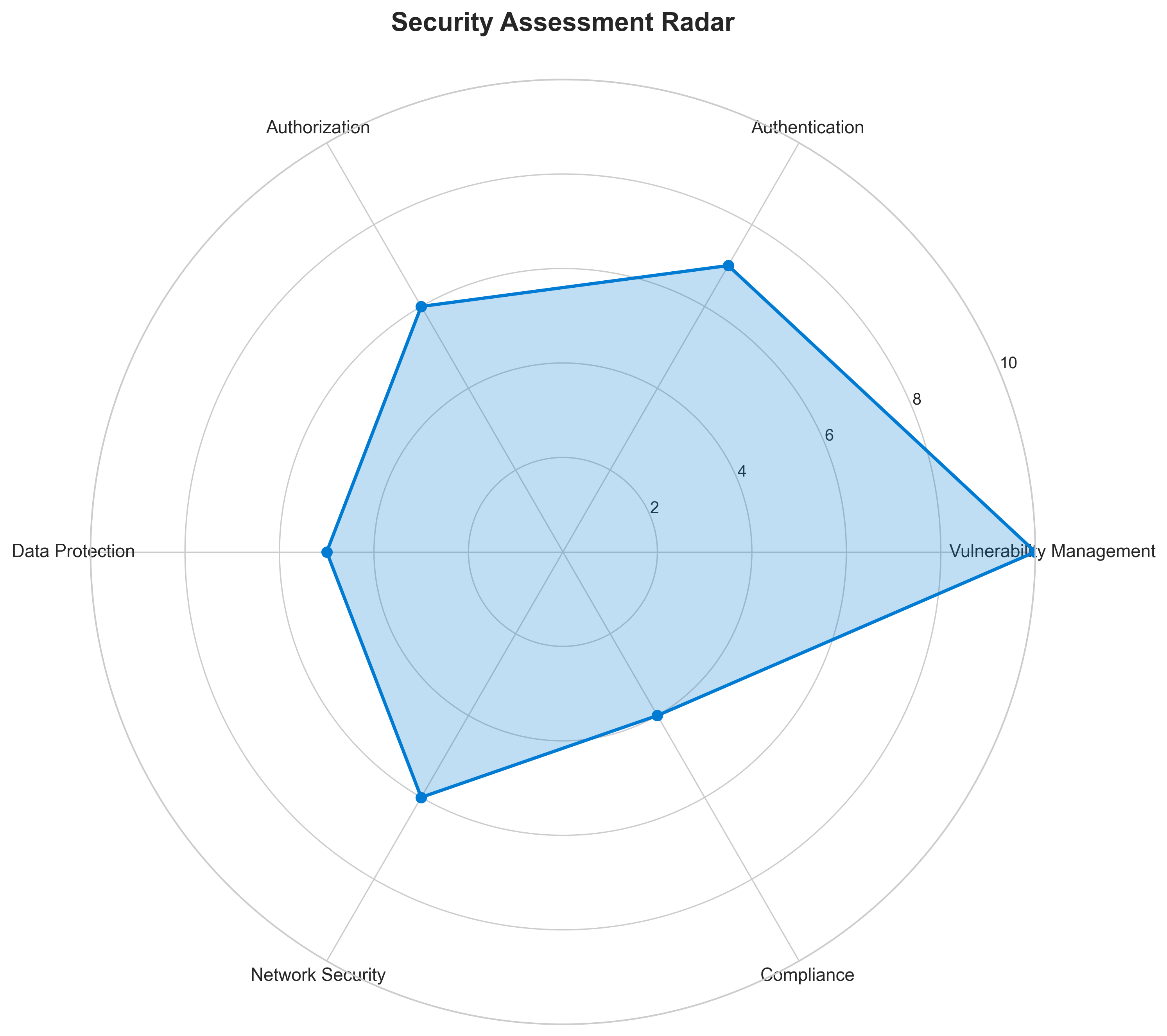
**📊 Business Impact:** Architecture supports unified deployment with architectural patterns need assessment considerations.

**📊 Recommendations:** Consider componentization for better scalability for improved maintainability.

**📊 Technical Details:** Architecture style: {'value': 'microservices', 'confidence': <ConfidenceLevel.HIGH: 'HIGH'>, 'evidence': ['Found 3 components', 'Multiple deployable components detected', '3 containerized components', 'Multiple deployment configurations'], 'reasoning': 'Multiple components with independent deployment characteristics', 'limitations': ['Cannot determine communication patterns without runtime analysis']}. Component count: 3. Complexity assessment: unknown.

────────────────────────────────────────────────────────────────────────────────

Security Assessment Radar



**📊 Context:** Security radar shows 3 security findings across application components.

**📊 Key Insights:** Security posture assessment reveals high risk profile with 3 base image vulnerabilities.

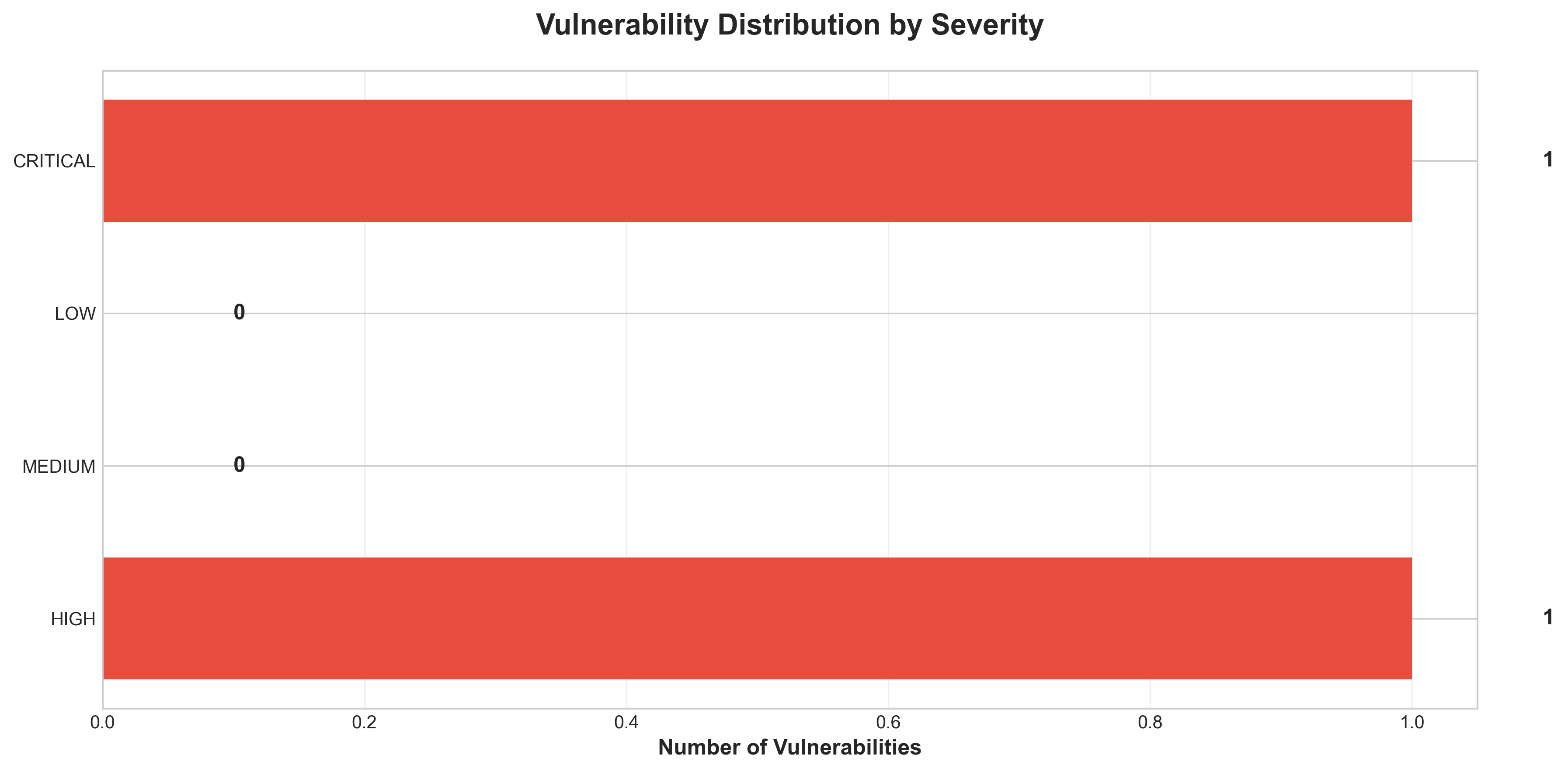
**📊 Business Impact:** Current security posture represents high risk to system reliability and compliance requirements.

**📊 Recommendations:** Prioritize security remediation - address critical findings immediately.

**📊 Technical Details:** Analysis includes 3 base image risks and code pattern analysis. Security score based on vulnerability severity.

────────────────────────────────────────────────────────────────────────────────

Vulnerability Analysis



**📊 Context:** This vulnerability timeline analysis, generated as part of a modernization and migration planning report, specifically details security vulnerabilities found within the application's components. It focuses on identifying critical and high-severity issues that require immediate attention to mitigate risks.

**📊 Key Insights:** The analysis reveals a significant reliance on outdated and end-of-life (EOL) base images, most notably the 'node:10-slim' image used by the 'result' component, which is flagged as CRITICAL. Additionally, the 'worker' component utilizes two HIGH severity base images ('openjdk:8-jre' and 'maven:3.5-jdk-8-alpine'), both known to contain vulnerabilities. A potential hardcoded secret in the 'voting-app' also contributes a HIGH severity finding. All 3 identified security findings are categorized as HIGH or CRITICAL severity.

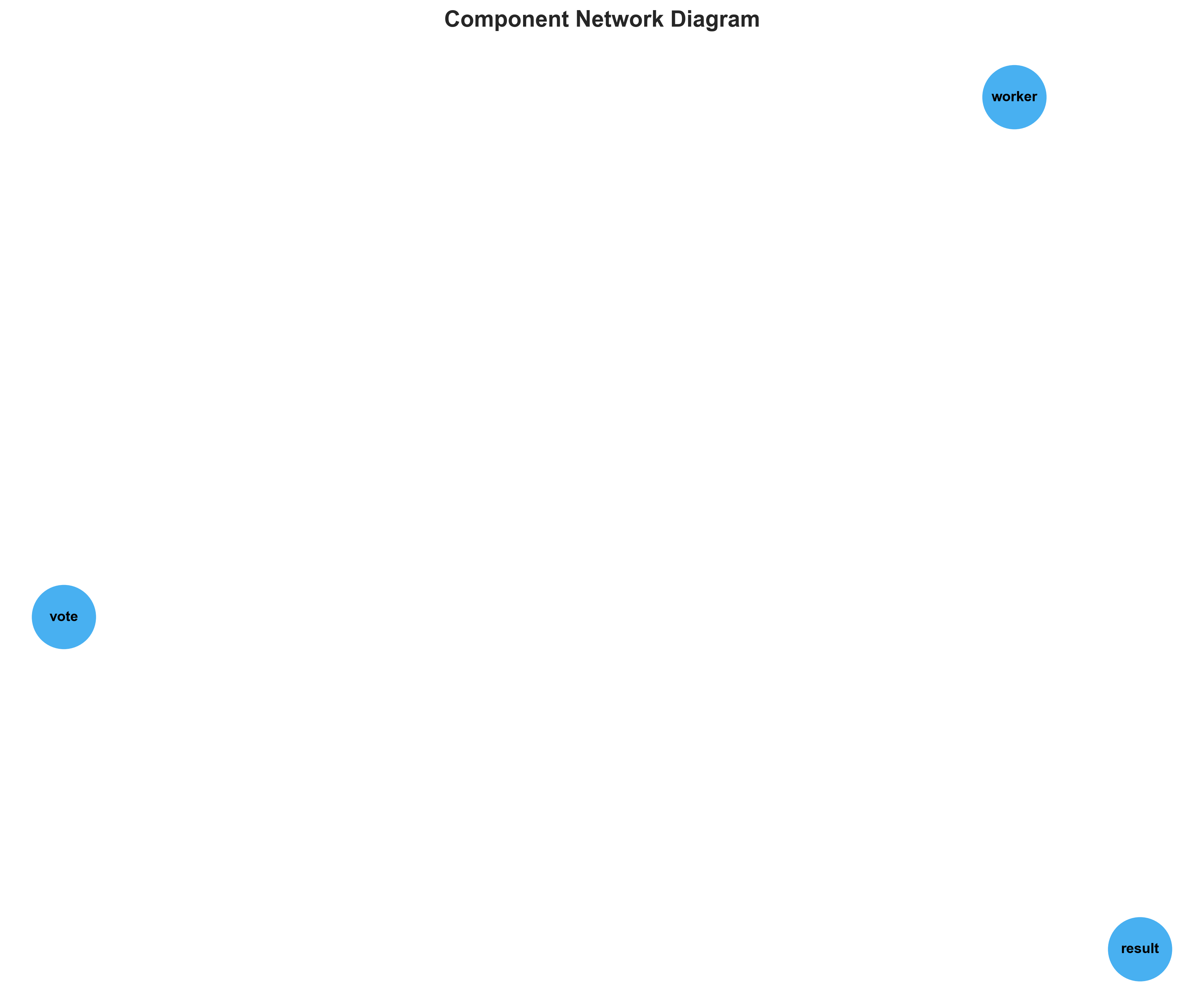
**📊 Business Impact:** The prevalence of EOL and vulnerable base images presents a substantial security risk, increasing the likelihood of exploitation and potential data breaches. This directly impacts the application's security posture, brand reputation, and compliance efforts. Addressing these issues is critical for a successful modernization and migration, as migrating a vulnerable application would carry forward these risks and potentially incur higher costs to remediate post-migration.

**📊 Recommendations:** Prioritize the immediate update of the 'result' component's base image from 'node:10-slim' to a supported version (e.g., node:18-slim or node:20-slim). Concurrently, address the HIGH severity vulnerabilities in the 'worker' component's base images ('openjdk:8-jre' and 'maven:3.5-jdk-8-alpine') by updating to recommended versions. Thoroughly review the 'voting-app' code to eliminate any hardcoded secrets.

**📊 Technical Details:** The findings originate from a base image analysis scan. The 'result' component's 'Dockerfile' at line 1 uses 'node:10-slim', which is identified as EOL and unsupported. The 'worker' component has issues with both its 'openjdk:8-jre' and 'maven:3.5-jdk-8-alpine' base images, contributing to the overall high severity count. The 'voting-app' has a specific code-level finding for a potential hardcoded secret.

────────────────────────────────────────────────────────────────────────────────

Component Network Topology



**📊 Context:** Shows status of 3 application components.

**📊 Key Insights:** 3 of 3 components are containerized. Fully ready for cloud deployment.

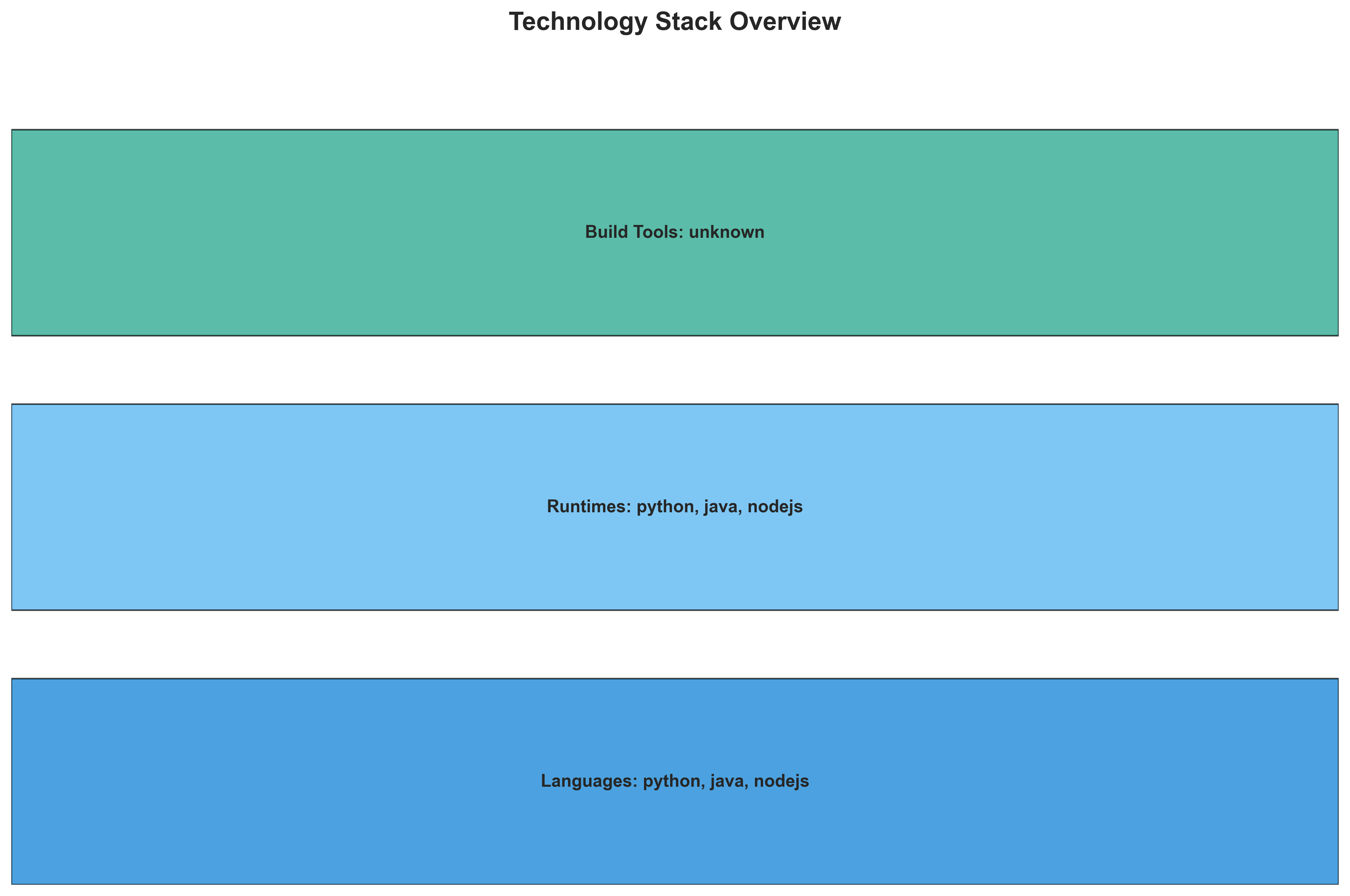
**📊 Business Impact:** Containerization status directly impacts cloud migration readiness and deployment flexibility.

**📊 Recommendations:** All components are containerized. Focus on optimizing container configurations and deployment strategies.

**📊 Technical Details:** Containerization rate: 3/3. Analysis includes Docker and orchestration configurations.

────────────────────────────────────────────────────────────────────────────────

Technology Stack



**📊 Context:** This diagram visualizes the technology stack of an application, detailing its constituent components, their primary languages, and containerization status. It serves as a foundational element for a comprehensive application intelligence report, specifically aimed at informing modernization and cloud migration strategies.

**📊 Key Insights:** The application comprises three distinct containerized components: 'vote' (Python), 'worker' (Java), and 'result' (Node.js). Notably, the 'worker' component utilizes outdated and potentially vulnerable base images ('openjdk:8-jre' and 'maven:3.5-jdk-8-alpine'), posing a significant security and stability risk. The presence of multiple base images for the 'worker' might indicate complex build processes or legacy configurations.

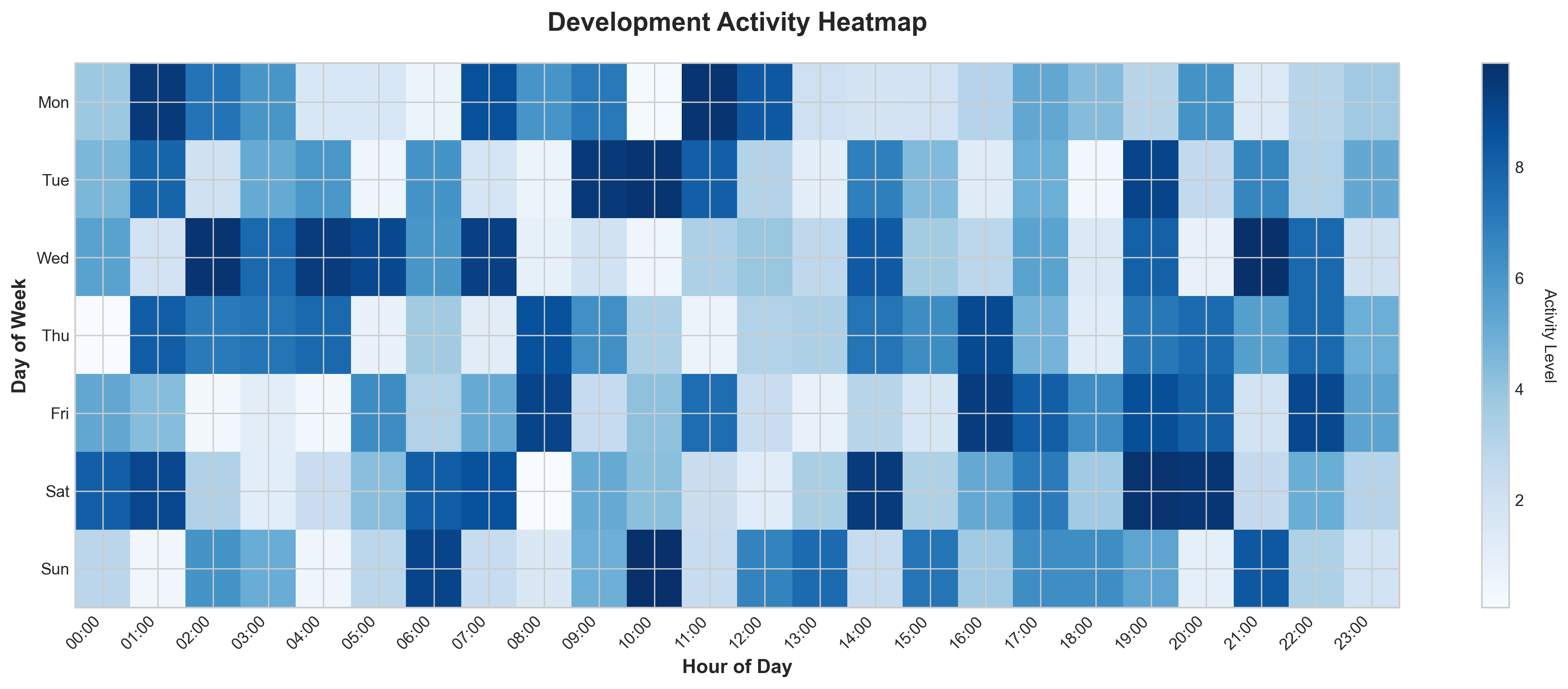
**📊 Business Impact:** The use of vulnerable base images in the 'worker' component presents a direct security risk, potentially exposing the application to exploits and data breaches. This necessitates immediate attention to mitigate these risks. The mixed language environment (Python, Java, Node.js) and containerization suggest a microservices-like architecture, which could offer flexibility for modernization but also introduces complexity in managing disparate technology versions and dependencies.

**📊 Recommendations:** Prioritize updating the 'worker' component's base images to current, supported, and secure versions of Java/OpenJDK and Maven to address immediate security vulnerabilities. Conduct a thorough dependency analysis for all components, particularly the 'worker', to identify and upgrade other outdated libraries. Investigate the 'worker's' multi-language implementation note to confirm the primary technology stack and plan accordingly for potential rationalization or consolidation.

**📊 Technical Details:** All identified components ('vote', 'worker', 'result') are confirmed to be containerized ('is\_containerized': true), utilizing Docker packaging. The 'vote' and 'result' components are built on lightweight base images ('python:3.9-slim', 'node:10-slim'), while 'worker' exhibits a combination of 'openjdk:8-jre' and 'maven:3.5-jdk-8-alpine'. The 'worker' component also specifies a JAVA\_APP\_JAR environment variable, indicating a Java application packaged as a JAR file.

────────────────────────────────────────────────────────────────────────────────

Development Activity Heatmap



**📊 Context:** This Development Activity Heatmap visualizes the current state of a 3-component application, detailing its technology stack, containerization status, external dependencies, and Git activity. The analysis is specifically designed to inform modernization and migration planning efforts, providing a technical baseline.

**📊 Key Insights:** The application comprises three distinct components, each containerized using Docker, and utilizes a polyglot stack (Java, Node.js, Python). It relies on four external services, including PostgreSQL and Redis (listed twice, indicating potential redundancy or aliasing). Notably, Git activity is minimal with only one commit and no active contributors recently, and all identified security findings are of HIGH severity, stemming from outdated base images in all components.

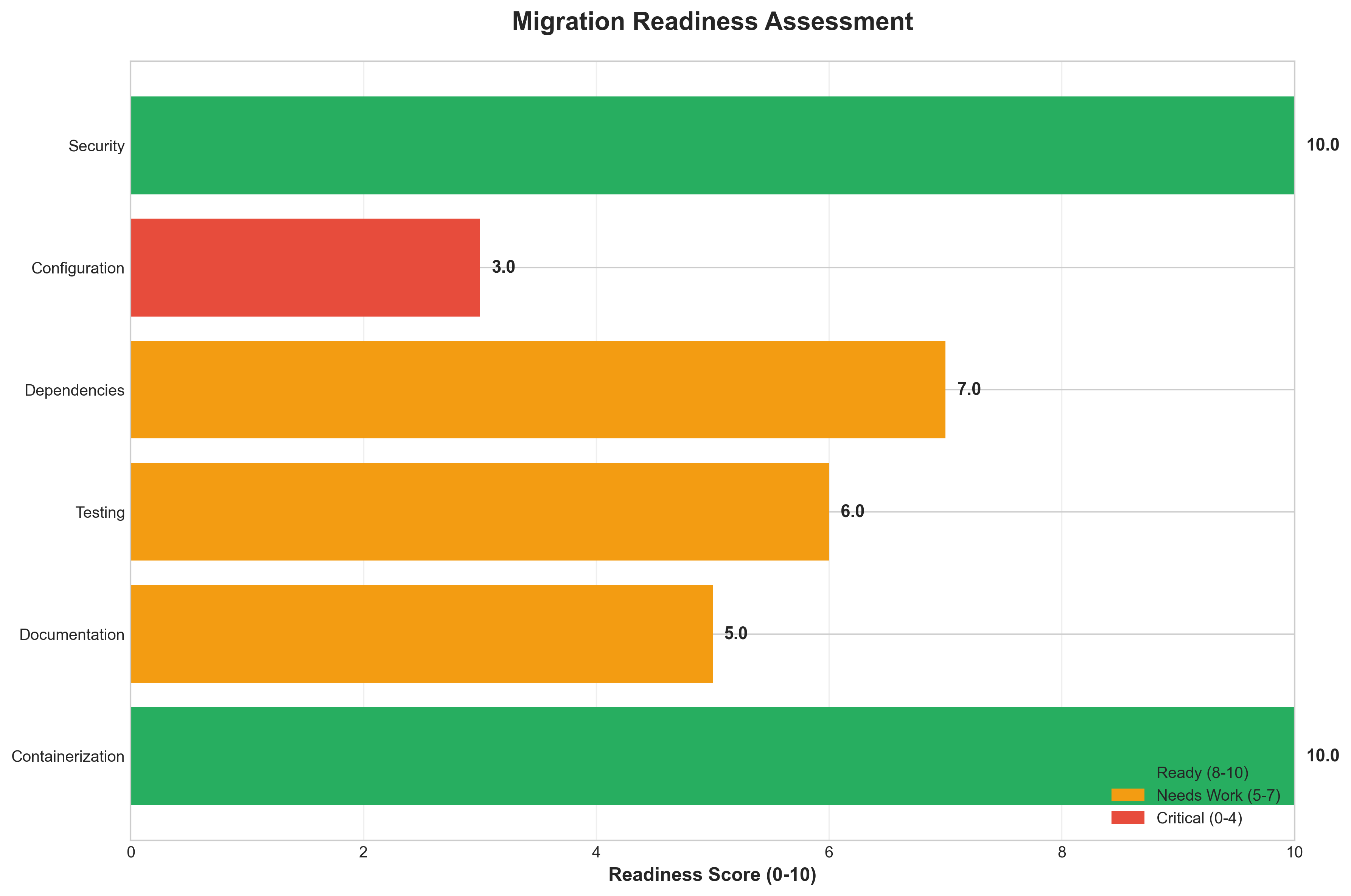
**📊 Business Impact:** The lack of recent development activity and the critical security vulnerabilities present a significant risk to business operations and continuity. The reliance on unpatched base images exposes the application to known exploits, potentially leading to data breaches or service disruptions. The modernization effort will require immediate attention to security patching, and the minimal Git history suggests a potential knowledge gap or lack of active stewardship, which could complicate migration timelines and increase associated costs.

**📊 Recommendations:** Prioritize immediate remediation of the HIGH severity security findings by updating all container base images to supported and patched versions. Conduct a thorough review of the Git history and contributor activity to assess current development capacity and expertise. Initiate a detailed technical assessment of the external service dependencies, particularly the duplicated Redis and PostgreSQL entries, to optimize infrastructure and reduce potential points of failure during migration.

**📊 Technical Details:** The application's architecture is identified as microservices with HIGH confidence, supported by three independent, containerized components. All components are containerized (3/3), with no explicit containerization files, Kubernetes resources, or Docker Compose files identified, suggesting a 'unknown' deployment platform and a lack of orchestration. The 'worker' component exhibits HIGH severity vulnerabilities in both its OpenJDK 8 and Maven base images, while the 'result' component shows similar issues with its Node.js 10 base image.

────────────────────────────────────────────────────────────────────────────────

Migration Readiness Assessment



**📊 Context:** This Migration Readiness Assessment focuses on the containerization status and security posture of an application slated for cloud migration. The analysis reveals critical vulnerabilities within the application's components, directly impacting its suitability for migration in its current state.

**📊 Key Insights:** The application exhibits a significant security risk, with all three identified components (worker and result) utilizing base images with HIGH severity vulnerabilities. Specifically, the 'worker' component relies on outdated OpenJDK 8 and Maven 3.5, while the 'result' component uses an End-of-Life Node.js 10 image. The containerization status score of 3 indicates a partial readiness, but these security flaws are a major impediment.

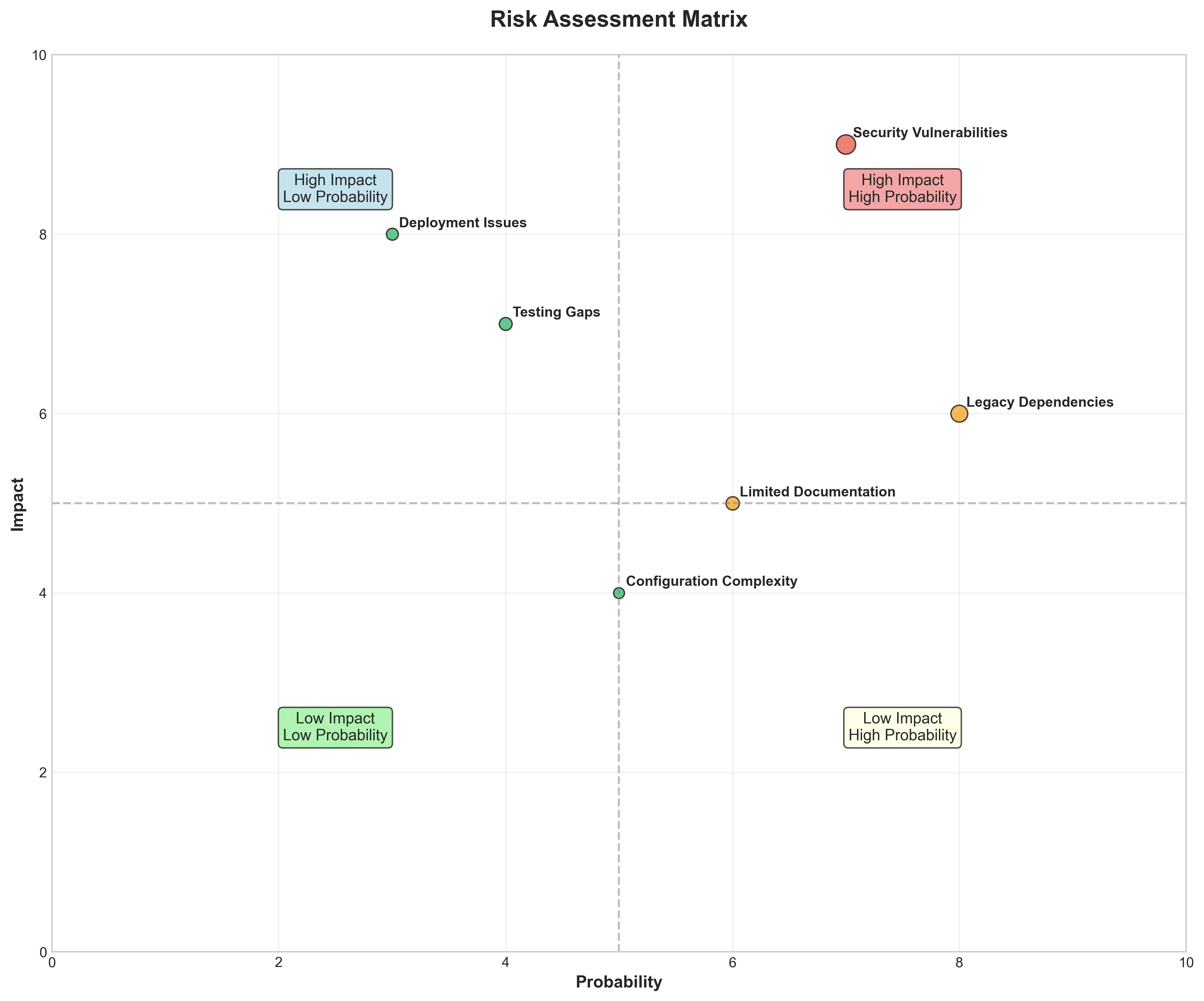
**📊 Business Impact:** The presence of critical vulnerabilities poses a substantial security risk to the business, potentially leading to data breaches, service disruptions, and compliance violations if migrated without remediation. This necessitates immediate action to address these security findings, which will likely impact the planned migration timeline and incur additional development costs for updating base images and potentially refactoring components.

**📊 Recommendations:** Prioritize the immediate remediation of all HIGH severity security findings by updating the base images for the 'worker' (openjdk:8-jre, maven:3.5-jdk-8-alpine) and 'result' (node:10-slim) components to actively maintained and secure versions. This proactive security patching is critical before proceeding with any cloud migration activities to mitigate risks.

**📊 Technical Details:** The security scan identified three HIGH severity findings across the application's three components. The 'worker' component utilizes 'openjdk:8-jre' and 'maven:3.5-jdk-8-alpine', both flagged for known vulnerabilities. The 'result' component uses 'node:10-slim', which is noted as End-of-Life and also contains numerous vulnerabilities.

────────────────────────────────────────────────────────────────────────────────

Risk Assessment Matrix



**📊 Context:** This Risk Assessment Matrix provides insights into application characteristics and technical architecture.

**📊 Key Insights:** Analysis of 3 components reveals system patterns and technical dependencies.

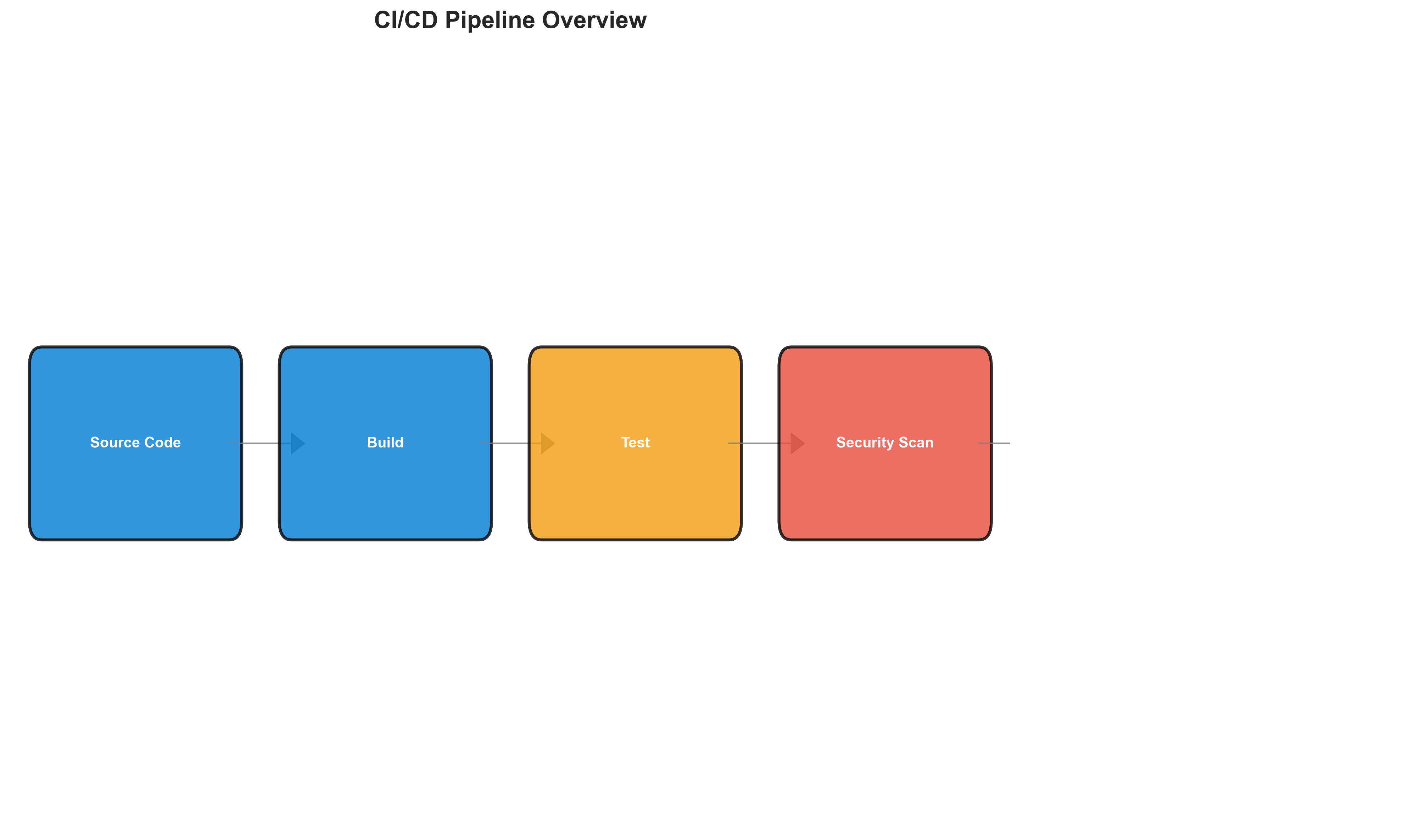
**📊 Business Impact:** Technical insights support strategic planning, risk assessment, and modernization decisions.

**📊 Recommendations:** Review detailed findings with technical teams to prioritize actions and plan next steps.

**📊 Technical Details:** Diagram type: Risk Assessment Matrix. Analysis includes component assessment, technology stack evaluation, and architectural patterns.

────────────────────────────────────────────────────────────────────────────────

CI/CD Pipeline Overview



**📊 Context:** This CI/CD pipeline overview visualizes a microservices architecture comprising three distinct components: Java, Node.js, and Python. The analysis is focused on informing modernization and migration strategies by highlighting the current state of the pipeline, its technology stack, and associated risks.

**📊 Key Insights:** The application exhibits a fully containerized microservices architecture, leveraging Docker for all three components, indicating a degree of maturity in deployment practices. However, significant security vulnerabilities have been identified within the base images of all components, with three high-severity findings related to outdated and unsupported Java and Node.js versions. Git history indicates minimal recent activity, suggesting potential stagnation or a shift in development focus.

**📊 Business Impact:** The presence of critical vulnerabilities in base images poses a significant security risk, potentially exposing the application and its data to breaches, which could lead to reputational damage and regulatory fines. The use of EOL (End-of-Life) Node.js versions exacerbates this risk. Failure to address these vulnerabilities could impede modernization efforts and increase the complexity and cost of any planned cloud migration.

**📊 Recommendations:** Prioritize updating all base container images to versions that are actively supported and free from known high-severity vulnerabilities. Conduct a thorough security review and remediation plan for the identified base image issues. Investigate the low Git commit count and lack of active contributors to understand current development velocity and potential technical debt accumulation.

**📊 Technical Details:** The pipeline supports Java, Node.js, and Python, with all three components packaged as Docker containers. External services include PostgreSQL and Redis (listed twice, suggesting a potential data quality issue or distinct instances). Notably, there are no identified data sources directly linked to the pipeline components, and the infrastructure summary reveals no orchestration files (Kubernetes, Docker Compose), indicating a 'deployment\_platform': 'unknown' status.

────────────────────────────────────────────────────────────────────────────────

Component Relationships (Graphviz)



**📊 Context:** Shows status of 3 application components.

**📊 Key Insights:** 3 of 3 components are containerized. Fully ready for cloud deployment.

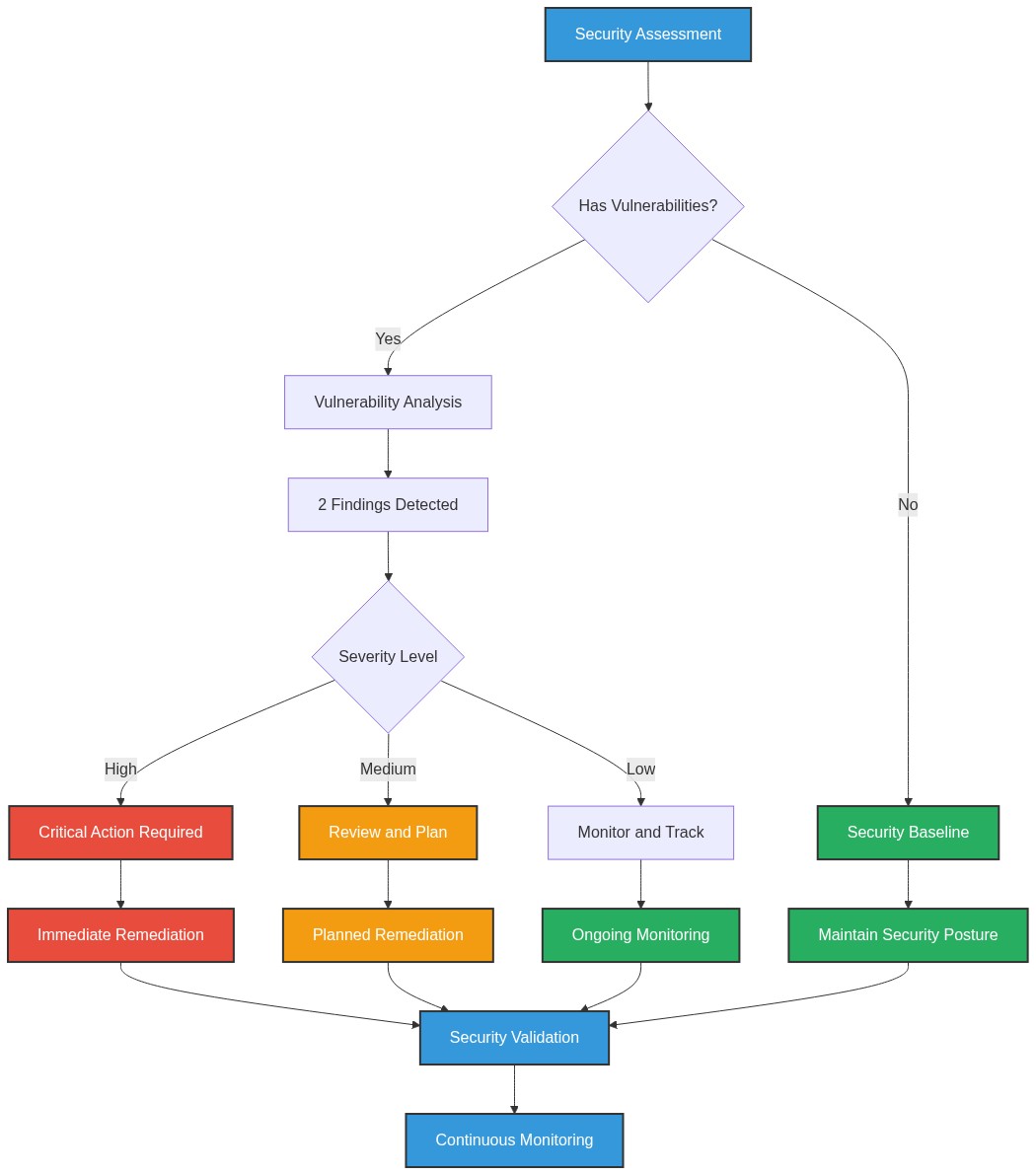
**📊 Business Impact:** Containerization status directly impacts cloud migration readiness and deployment flexibility.

**📊 Recommendations:** All components are containerized. Focus on optimizing container configurations and deployment strategies.

**📊 Technical Details:** Containerization rate: 3/3. Analysis includes Docker and orchestration configurations.

────────────────────────────────────────────────────────────────────────────────

Security Flow Diagram (Mermaid)



**📊 Context:** Security radar shows 3 security findings across application components.

**📊 Key Insights:** Security posture assessment reveals high risk profile with 3 base image vulnerabilities.

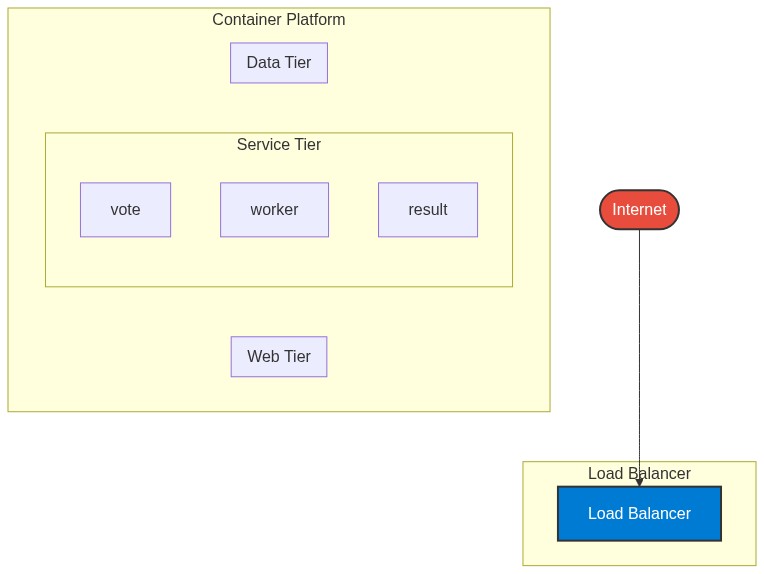
**📊 Business Impact:** Current security posture represents high risk to system reliability and compliance requirements.

**📊 Recommendations:** Prioritize security remediation - address critical findings immediately.

**📊 Technical Details:** Analysis includes 3 base image risks and code pattern analysis. Security score based on vulnerability severity.

────────────────────────────────────────────────────────────────────────────────

Deployment Architecture (Mermaid)



**📊 Context:** Architecture diagram shows {'value': 'microservices', 'confidence': <ConfidenceLevel.HIGH: 'HIGH'>, 'evidence': ['Found 3 components', 'Multiple deployable components detected', '3 containerized components', 'Multiple deployment configurations'], 'reasoning': 'Multiple components with independent deployment characteristics', 'limitations': ['Cannot determine communication patterns without runtime analysis']} pattern with 3 components.

**📊 Key Insights:** System follows {'value': 'microservices', 'confidence': <ConfidenceLevel.HIGH: 'HIGH'>, 'evidence': ['Found 3 components', 'Multiple deployable components detected', '3 containerized components', 'Multiple deployment configurations'], 'reasoning': 'Multiple components with independent deployment characteristics', 'limitations': ['Cannot determine communication patterns without runtime analysis']} architecture with unknown operational complexity. Offers requires further analysis.

**📊 Business Impact:** Architecture supports unified deployment with architectural patterns need assessment considerations.

**📊 Recommendations:** Consider componentization for better scalability for improved maintainability.

**📊 Technical Details:** Architecture style: {'value': 'microservices', 'confidence': <ConfidenceLevel.HIGH: 'HIGH'>, 'evidence': ['Found 3 components', 'Multiple deployable components detected', '3 containerized components', 'Multiple deployment configurations'], 'reasoning': 'Multiple components with independent deployment characteristics', 'limitations': ['Cannot determine communication patterns without runtime analysis']}. Component count: 3. Complexity assessment: unknown.

────────────────────────────────────────────────────────────────────────────────

Risk Assessment Flow (Mermaid)



**📊 Context:** This Risk Assessment Flow provides insights into application characteristics and technical architecture.

**📊 Key Insights:** Analysis of 3 components reveals system patterns and technical dependencies.

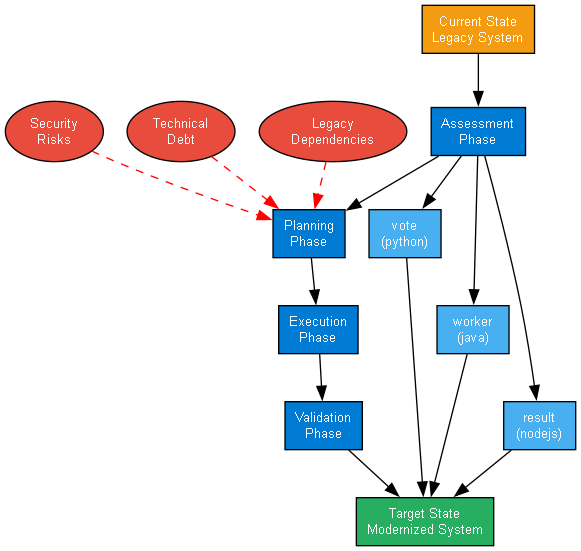
**📊 Business Impact:** Technical insights support strategic planning, risk assessment, and modernization decisions.

**📊 Recommendations:** Review detailed findings with technical teams to prioritize actions and plan next steps.

**📊 Technical Details:** Diagram type: Risk Assessment Flow. Analysis includes component assessment, technology stack evaluation, and architectural patterns.

────────────────────────────────────────────────────────────────────────────────

Migration Strategy (Graphviz)



**📊 Context:** Migration strategy analysis for 3 components shows requires preparation cloud readiness.

**📊 Key Insights:** System is 100% containerized with 3 security concerns. Migration complexity: requires preparation.

**📊 Business Impact:** Migration timeline estimated at 6+ months with requires preparation current readiness state.

**📊 Recommendations:** Priority actions: Address containerization gaps and security issues before migration.

**📊 Technical Details:** Readiness factors: 3/3 containerized, 3 security findings. Strategy: lift-and-shift feasible for containerized components.

────────────────────────────────────────────────────────────────────────────────