

# Code Migration Tool

---

An AI-powered migration accelerator for modernizing legacy systems with greater confidence


LOWER RISK

BETTER CONTROL

FASTER OUTCOME

# Why Code Migration Needs a Smarter Approach


Code migration is more than code conversion, it requires managing complexity, risk, validation, and business impact



### Compatibility Gaps

Legacy and modern systems often have incompatible architectures and design patterns.


▲ High Risk



### Bugs & Performance Risk

Unexpected bugs and performance degradation emerge


▬ Performance



### Cost & Time Pressure

High cost and significant time demand for migration


● 60% Over



### Testing & Validation

Heavy effort for testing, validation, and retraining


▽ Critical



### Framework Complexity

Higher migration risk across languages and frameworks

● Complex



### Key Insight

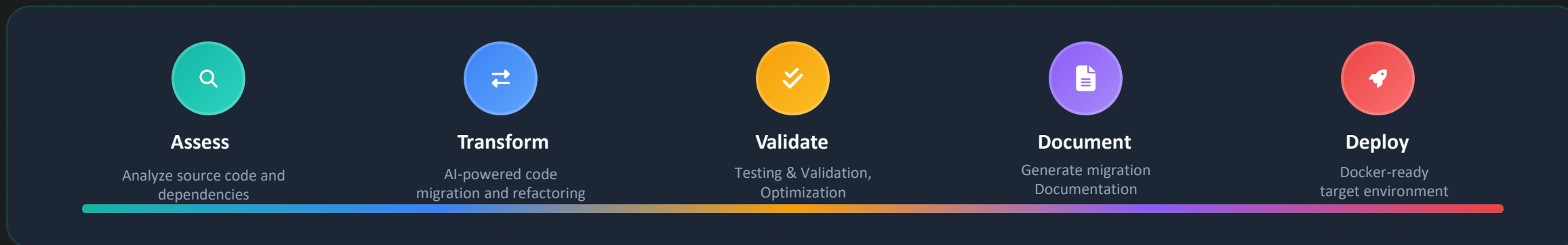
Migration affects timelines, cost, quality, validation effort, and business confidence across the entire lifecycle

85%  
Projects Delayed

40%  
Over Budget

# How the CMT Works

A structured, AI-powered migration workflow designed to reduce complexity and improve modernization outcomes.



## AI-Powered Code Translations

Automatically transforms legacy code into the target environment with reduced manual effort.

 Translate

## Dedicated Dependency Analyzer

Analyzes source-code dependencies to identify migration gaps and complexity early.

 Analyze


## Test-driven validation

Supports validation through testing to improve quality, stability, and confidence.

 Validate

## Optimization Support

Helps identify improvement opportunities for better performance and modernization outcomes.

 Document

## Native code migration support

Supports migration across programming languages, frameworks, and target architectures

 Deploy

## Seamless Docker integration

Enables Docker-based rebuilding and deployment for a more controlled migration process.

 Deploy

# Value and Business Impact

Business outcomes delivered through the accelerator through higher productivity, faster modernization, and greater confidence.

**60%**

## Productivity Gain

Compared to traditional migration methods

**Up to 60% Faster**



**85%**

## Lower Migration Risk

Reduce compatibility gaps, defects, and modernization uncertainty



**3x**

## Faster Time-to-Market

Accelerate transformation through automated analysis and migration



**40%**

## Better Cost Efficiency

Reduce manual effort and improve modernization productivity



**99%**

## Stronger Validation Confidence

Support quality through testing, validation, and optimization

**85%**

Risk Reduction

**3x**

Speed Improvement



## Key Business Impact

CMT helps organizations modernize faster with stronger confidence in outcomes

**Scalable**

**Predictable**

Dashboard

Refactoring Migration Unit Testing Functional Testing

History results + New Migration

Sr. No.	Project Name	Date	Step	Status	Action	Download
1	CMT212-nodejs-2.0-Java-v1-v13	2025-01-15	Migration	Completed	Proceed to Unit Test	Download
2	CMT212-nodejs-2.0-Java-v1-v13	2025-01-14	Unit Testing	In Progress	Rerun	Download
3	CMT212-nodejs-2.0-Java-v1-v13	2025-01-13	Migration	In Progress	Rerun	Download
4	CMT212-nodejs-2.0-Java-v1-v13	2025-01-12	Refactoring	Completed	Proceed to Unit Test	Download
5	CMT212-nodejs-2.0-Java-v1-v13	2025-01-10	Functional Testing	Pending	Rerun	Download

Proclink • Code Migration Tool © 2025 Proclink. All rights reserved.

## 1. Dashboard Overview

## 2. Migration Configuration

← Back to Dashboard

### Migration Configuration

Configure your source and target environments for automated unit testing

#### Source Directory

Project Name

Source Path

Source Language

Source Version (Optional)

Source Framework

Source Entry Files

Source Test Files

Source URL

#### Target Directory

Target Directory

Target Language

Target Version

Target Framework

Target URL

Step

#### Migration Summary

Selected Files: 0

Estimated Duration: ~15 min

**Migration Progress**  
Automated test generation and execution with AI-powered recommendations

**Migration Steps**

- Step 1: Environment Setup
  - ✓ Create target directory
  - ✓ Repair isolated environment for target language
- Step 2: Analyze Dependencies
  - ✓ Identify project libraries, frameworks, and external services
  - ✓ Check compatibility and required replacements
- Step 3: Code Migration Progress
  - demo\_project/app\_main.py
  - Progress: 60%
- Step 4: Identifying Independent Files & Copy in Target
  - Move them to the target project structure
  - Detect static or conversion-free files
- Step 5: Code Migration Status & Final Report
  - Generate migration summary report

**Migration Execution Log**

```
001 Initializing unit test suite...
002 Connecting to test server...
003 Server validation successful
004 Scanning migrated code for testable units...
005 Generating test cases for App.tsx
006 Generating test cases for helpers.ts
```

## 3. Migration Progress

## 4. Token Details

**Migration Token Details**  
Review the estimated token usage and cost for this migration

Total Input Tokens	45,230	Total Output Tokens	32,150
Number of Iterations	3	Estimated Total Cost	-

Cancel | Proceed to Migration