Migration Toolkit for Runtimes 1.1

Introduction to the Migration Toolkit for Runtimes

Learn how to use the Migration Toolkit for Runtimes to migrate and modernize Java applications and components.

Last Updated: 2023-06-27

Migration Toolkit for Runtimes 1.1 Introduction to the Migration Toolkit for Runtimes

Learn how to use the Migration Toolkit for Runtimes to migrate and modernize Java applications and components.

Legal Notice

Abstract

This guide provides information to help you get started with the Migration Toolkit for Runtimes.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

CHAPTER 1. INTRODUCTION

1.1. ABOUT THE INTRODUCTION TO THE MIGRATION TOOLKIT FOR RUNTIMES

This guide is for engineers, consultants, and others who want to use the Migration Toolkit for Runtimes (MTR) to migrate Java applications or other components. It provides an overview of the Migration Toolkit for Runtimes and how to get started using the tools to plan and run your migration.

CHAPTER 2. ABOUT THE MIGRATION TOOLKIT FOR RUNTIMES

What is the Migration Toolkit for Runtimes?

The Migration Toolkit for Runtimes (MTR) is an extensible and customizable rule-based tool that simplifies the migration and modernization of Java applications.

MTR examines application artifacts, including project source directories and application archives, and then produces an HTML report highlighting areas needing changes. MTR supports many migration paths, including the following examples:

- Upgrading to the latest release of Red Hat JBoss Enterprise Application Platform
- Migrating from Oracle WebLogic or IBM WebSphere Application Server to Red Hat JBoss Enterprise Application Platform
- Containerizing applications and making them cloud-ready
- Migrating from Java Spring Boot to Quarkus
- Updating from Oracle JDK to OpenJDK
- Upgrading from OpenJDK 8 to OpenJDK 11
- Upgrading from OpenJDK11 to OpenJDK 17
- Migrating EAP Java applications to Azure
- Migrating Spring Boot Java applications to Azure

For more information about use cases and migration paths, see the MTR for developers web page.

How does the Migration Toolkit for Runtimes simplify migration?

The Migration Toolkit for Runtimes looks for common resources and known trouble spots when migrating applications. It provides a high-level view of the technologies used by the application.

MTR generates a detailed report evaluating a migration or modernization path. This report can help you to estimate the effort required for large-scale projects and to reduce the work involved.

2.1. MTR FEATURES

The Migration Toolkit for Runtimes (MTR) provides a number of capabilities to assist with planning and executing migration projects.

Planning and work estimation

MTR assists project managers by detailing the type of work and estimation of effort to complete the tasks. Level of effort is represented in MTR reports as story points. Actual estimates will be based on the skills required and the classification of migration work needed.

Identifying migration issues and providing solutions

MTR identifies migration issues and highlights specific points in the code where an issue occurs. MTR suggests code changes and provides additional resources to help engineers resolve the specific issue.

Detailed reporting

MTR produces numerous reports to give both high-level views of the migration effort and details of specific migration tasks. You can view migration issues across all applications, charts and summary information about issues in an application, a breakdown of issues by module in the application, reports of technologies used, and dependencies on other applications and services. You can also examine source files to see the line of code where an issue occurs. See the *CLI Guide* for more information on the available MTR reports.

Built-in rules and migration paths

MTR comes with a core set of rules to provide migration assistance for several common migration paths. These rules identify the use of proprietary functionality from other application servers or deprecated subsystems from previous versions of JBoss EAP. MTR also contains rules to identify common migration issues, such as hard-coded IP addresses and JNDI lookups.

Rule extensibility and customization

MTR provides the ability to create powerful and complex rules. You can expand upon the core set of rules provided by MTR and create rules to identify additional issues that are important for your migration project. You can also override core rules and create custom rule categories. See the *Rules Development Guide* for more information on customizing MTR rules.

Ability to analyze source code or application archives

MTR can evaluate application archives or source code, and can evaluate multiple applications together. It can identify archives that are shared across multiple applications, which can help with more accurate effort estimation.

2.2. ABOUT MTR RULES

The Migration Toolkit for Runtimes (MTR) contains rule-based migration tools that analyze the APIs, technologies, and architectures used by the applications you plan to migrate. In fact, the MTR analysis process is implemented using MTR rules. MTR uses rules internally to extract files from archives, decompile files, scan and classify file types, analyze XML and other file content, analyze the application code, and build the reports.

MTR builds a data model based on the rule execution results and stores component data and relationships in a graph database, which can then be queried and updated as needed by the migration rules and for reporting purposes.

MTR rules use the following rule pattern:

```
when(condition)
  perform(action)
otherwise(action)
```

MTR provides a comprehensive set of standard migration rules out-of-the-box. Because applications may contain custom libraries or components, MTR allows you to write your own rules to identify use of components or software that may not be covered by the existing ruleset.

If you plan to write your own custom rules, see the *Rules Development Guide* for detailed instructions.

CHAPTER 3. SUPPORTED CONFIGURATIONS

3.1. SUPPORTED MIGRATION PATHS

The Migration Toolkit for Runtimes (MTR) supports migrations from third-party enterprise application servers, such as Oracle WebLogic Server, to JBoss Enterprise Application Platform (JBoss EAP) and upgrades to the latest release of JBoss EAP.

MTR provides a comprehensive set of rules to assess the suitability of your applications for containerization and deployment on Red Hat OpenShift Container Platform (RHOCP). You can run an MTR analysis to assess your applications' suitability for migration to multiple target platforms.

The following table describes the migration paths most commonly supported.

Table 3.1. Supported migration paths: Source platform ⇒ target

Source platform ⇒	JBoss EAP 6	JBoss EAP 7	RHOCP	OpenJDK 8, 11, & 17	Apache Camel 3	Spring Boot on Red Hat Runtimes	Quarkus	Azure
Oracle WebLogic Server	✔	✔	✔	✔	-	-	-	-
IBM WebSphere Application Server	✔	✔	✔	✔	-	-	-	-
JBoss EAP 4	✔	✘ [a]	✔	✔	-	-	-	-
JBoss EAP 5	✔	✔	✔	✔	-	-	-	-
JBoss EAP 6	N/A	✔	✔	✔	-	-	-	-
JBoss EAP 7	N/A	✔	✔	✔	-	-	-	✔
Oracle JDK	-	-	✔	✔	-	-	-	-
Apache Camel 2	-	-	✔	✔	✔	-	-	-
Spring Boot	-	-	✔	✔	-	✔	✔	✔
Java application	-	-	✔	✔	-	-	-	-

[a] Although MTR does not currently provide rules for this migration path, Red Hat Consulting can assist with migration from any source platform to JBoss EAP 7.

CHAPTER 4. PREREQUISITES

- Java Development Kit (JDK) installed. MTR supports the following JDKs:
  - OpenJDK 11
  - OpenJDK 17
  - Oracle JDK 11
  - Oracle JDK 17
  - Eclipse Temurin™ JDK 11
  - Eclipse Temurin™ JDK 17
- 8 GB RAM
- macOS installation: the value of **maxproc** must be **2048** or greater.

CHAPTER 5. ABOUT THE TOOLS

The Migration Toolkit for Runtimes (MTR) provides several tools to assist you in the various stages of your migration and modernization efforts. Review the details of each tool to determine which one is right for your project.

- Web console
- Migration Toolkit for Runtimes Operator
- CLI
- IDE addons for:
  - Eclipse and Red Hat CodeReady Studio
  - Visual Studio Code, Visual Studio Codespaces, and Eclipse Che
  - IntelliJ IDEA
- Maven plugin

5.1. ABOUT THE CLI

The CLI is a command-line tool in the Migration Toolkit for Runtimes that allows you to assess and prioritize migration and modernization efforts for applications. It provides numerous reports that highlight the analysis without the overhead of the other tools. The CLI includes a wide array of customization options, and allows you to finely tune MTR analysis options or integrate with external automation tools.

For more information about using the CLI, see the MTR *CLI Guide*.

5.2. ABOUT THE WEB CONSOLE

The web console for the Migration Toolkit for Runtimes allows a team of users to assess and prioritize migration and modernization efforts for a large number of applications. It allows you to group applications into projects for analysis and provides numerous reports that highlight the results.

5.3. ABOUT THE MTR PLUGIN FOR ECLIPSE AND RED HAT CODEREADY STUDIO

The Migration Toolkit for Runtimes (MTR) plugin for Eclipse and Red Hat CodeReady Studio helps you migrate and modernize applications.

The MTR plugin analyzes your projects using customizable rulesets, marks migration issues in the source code, provides guidance to fix the issues, and offers automatic code replacement, or Quick Fixes, if possible.

For more information on using the MTR plugin, see the MTR *Eclipse and Red Hat CodeReady Studio Guide*.

5.4. ABOUT THE MTR EXTENSION FOR VISUAL STUDIO CODE

The Migration Toolkit for Runtimes (MTR) extension for Visual Studio Code helps you migrate and modernize applications.

The MTR extension is also compatible with Visual Studio Codespaces, the Microsoft cloud-hosted development environment.

The MTR extension analyzes your projects using customizable rulesets, marks issues in the source code, provides guidance to fix the issues, and offers automatic code replacement, if possible.

For more information about using the MTR extension, see the MTR *Visual Studio Code Extension Guide*.

5.5. ABOUT THE MAVEN PLUGIN

The Maven plugin for the Migration Toolkit for Runtimes integrates into the Maven build process, allowing developers to continuously evaluate migration and modernization efforts with each iteration of source code. It provides numerous reports that highlight the analysis results, and is designed for developers who want updates with each build.

CHAPTER 6. PLANNING YOUR APPLICATION MIGRATION

6.1. GOALS OF ASSESSING A MIGRATION

Many organizations are faced with the challenges of keeping their existing business operations running, while also trying to innovate. There are typically increased expectations to deliver new functionality faster and to reduce cost.

When assessing a migration or modernization effort, it is important to address the challenges that are specific to your organization. Common challenges that organizations face are limited budgets, lack of in-house skills, perceived risks, no known predictable process, and accurate effort estimation in a timely manner.

In general, the goals when assessing a large-scale middleware migration or modernization effort are the following:

- Predicting the level of effort and cost
- Scheduling application migrations and handling conflicts
- Identifying all potential risks at a code, infrastructure, process, or knowledge level
- Predicting the return on investment to make the business case
- Identifying and mitigating risks to the business
- Minimizing disruption to existing business operations

Red Hat has developed a strategy to minimize risk and make application migration and modernizations more efficient.

6.2. RED HAT'S APPLICATION MIGRATION APPROACH

Red Hat has defined a strategy to make large-scale application migrations to Red Hat JBoss Middleware quantifiable, less costly, less risky, and easier to complete.

Migrating from proprietary or outdated middleware platforms to Red Hat JBoss Middleware gives you lightweight, modular, and cloud-ready middleware with a supporting application infrastructure to make teams more productive. JBoss Middleware and other Red Hat technologies also provide great opportunities to modernize your application development and delivery, allowing you to innovate more quickly.

6.2.1. Best practices

Red Hat recommends the following practices when planning and executing a migration or modernization:

- Create a centralized collaboration platform for information sharing.
  Comprehensive and accessible documentation is important so that knowledge can be easily shared and to ensure that effort is not duplicated by solving the same issue twice. It is recommended to document the following items:
  - A step-by-step guide for migrating or modernizing an application from scratch.
  - A collection of solutions to known and encountered issues.

    - Information about the new platform.
    - A record of the changes made to specific pilot projects.

- Reuse, automate, and standardize as much as possible.
  Consider reusing existing components instead of creating new ones. Automate processes related to the application lifecycle, such as builds, configuration, deployment, and tests. Standardize and document software packaging format, configuration management, libraries, and dependencies.

- Use a proven and repeatable methodology.
  The recommendation is to follow a practical approach and make as few changes as possible to get a functionally identical application.

- Involve Red Hat technical expertise early for the chosen Red Hat Middleware components.
  This is crucial to make your migrations and modernizations low risk, predictable, and efficient. Contact Red Hat Consulting for assistance.

6.2.2. Methodology

Red Hat's recommended methodology is a proven, scalable approach that helps you to incrementally plan and conduct a migration or modernization.

Figure 6.1. Red Hat migration methodology

The approach consists of the following phases:

Discover

Explore the technologies and identify concerns, organization requirements and challenges. Discuss options and potential approaches.

Design

Identify and mitigate the biggest risks from an application, infrastructure, process, and knowledge perspective. Establish a strategy for the solution. Prove the technology and build the business case.

Deploy

Scale the modernization or migration iteratively by executing the previously defined strategy according to a well-defined delivery model.

6.2.2.1. Discover phase

Figure 6.2. Red Hat migration methodology: Discover phase

The *Discover* phase is when you gather all stakeholders and decision-makers together to understand the current state and business drivers, and to determine the migration or modernization needs.

In this phase, you can do the following:

- Explore technologies and discuss potential approaches.
- Identify the existing pain points, concerns, requirements, and some potential challenges.
- Define the high-level business priorities, and scope the assessment.
- Determine in what ways you want to modernize your application development and delivery to allow you to innovate more rapidly.

Typically, this can be covered in a day in a workshop with Red Hat experts.

6.2.2.2. Design phase

Figure 6.3. Red Hat migration methodology: Design phase

The *Design* phase is when you identify all risks, define a strategy and target architecture, prove the technology, and build the business case. This phase consists of the following steps:

Assess

Examine the existing infrastructure, architecture, technologies, and applications. Identify dependencies, knowledge, processes, and lifecycles. Define the desired infrastructure, architecture, technologies, and applications. Determine the feasibility and potential risks. Draft a plan and provide rough estimates.
Analyzing applications using the Migration Toolkit for Runtimes web console or CLI helps to determine dependencies, potential risks, and relative effort. See the *Web Console Guide* or *CLI Guide* for information on how to use these tools.

Prove

Solve and document the identified technical risks, for example, high risk items and issues with unknown effort. Build and shape the new platform infrastructure. Refine the estimates based on the outcomes.

Pilot

Choose a small set of representative applications to transform. Finalize the target platform infrastructure as necessary. Update the documentation as the process is fine-tuned. This pilot consists of one or several iterations of the migration execution, which is scaled during the Deploy phase.
Use the Migration Toolkit for Runtimes MTR plugin to accelerate the code migration. See the *Eclipse and Red Hat CodeReady Studio Guide* for information on how to use the MTR plugin.

Plan

Based on the outcomes of the previous steps, sharpen the estimates and refine the project plan. Define the rollout strategy to be used in the Deploy phase to complete the migration. Prepare and schedule the relevant technical enablement courses.

6.2.2.3. Deploy phase

Figure 6.4. Red Hat migration methodology: Deploy phase

The *Deploy* phase is when you run the plan created in the Design phase. In this phase, you scale the overall transformation process to complete the plan and bring all applications to their new production environment.

The plan is run in iterations to deliver value incrementally. With each iteration, you continuously validate against the plan and document findings to improve the next sprint.

The use of the Migration Toolkit for Runtimes MTR plugin increases speed in each iteration. It can be used with Eclipse or Red Hat CodeReady Studio, and marks migration issues directly in the source code, provides inline hints, and offers code change suggestions. See the *Eclipse and Red Hat CodeReady Studio Guide* for information on how to use the MTR plugin.

APPENDIX A. REFERENCE MATERIAL

A.1. ADDITIONAL RESOURCES

A.1.1. Getting involved

To help the Migration Toolkit for Runtimes cover most application constructs and server configurations, including yours, you can help with any of the following items.

- Send an email to jboss-migration-feedback@redhat.com and let us know what MTR migration rules should cover.
- Provide example applications to test migration rules.
- Identify application components and problem areas that may be difficult to migrate.
  - Write a short description of these problem migration areas.
  - Write a brief overview describing how to solve the problem migration areas.
- Try Migration Toolkit for Runtimes on your application. Be sure to report any issues you encounter.
- Contribute to the Migration Toolkit for Runtimes rules repository.
  - Write a Migration Toolkit for Runtimes rule to identify or automate a migration process.
  - Create a test for the new rule.
  - Details are provided in the *Rules Development Guide*.
- Contribute to the project source code.
  - Create a core rule.
  - Improve MTR performance or efficiency.

Any level of involvement is greatly appreciated!

A.1.2. Resources

- MTR forums: https://developer.jboss.org/en/windup
- Jira issue tracker: https://issues.redhat.com/projects/WINDUP
- MTR mailing list: jboss-migration-feedback@redhat.com

A.1.3. Reporting issues

MTR uses Jira as its issue tracking system. If you encounter an issue executing MTR, submit a Jira issue.

Revised on 2023-06-27 21:22:01 UTC