Migration Toolkit for Applications 7.0

用户界面指南

使用 Migration Toolkit for Applications 用户界面将您的应用程序分组到项目中进行分析。

Last Updated: 2024-05-24

Migration Toolkit for Applications 7.0 用户界面指南

使用 Migration Toolkit for Applications 用户界面将您的应用程序分组到项目中进行分析。

法律通告

摘要

本指南介绍了如何在 Red Hat OpenShift 的混合云环境中使用 Migration Toolkit for Applications 用户界面来加快大规模应用程序现代化工作。

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。有关更多详情，请参阅我们的首席技术官 Chris Wright 提供的消息。

第 1 章 简介

1.1. 关于用户界面指南

本指南适用于希望使用 Migration Toolkit for Applications (MTA) 用户界面在 Red Hat OpenShift 的混合云环境中加速大型应用程序现代化工作的架构师、工程师、顾问和其他人员。此解决方案会考虑整个迁移过程的详细情况，包括产品组合和应用程序级别的信息：库存、评估、分析和管理应用程序，以便更快地通过用户界面迁移到 OpenShift。

注意

Migration Toolkit for Applications 5 提供的迁移解决方案，*x* 版本（迁移和 Java 应用程序现代化）随运行时 1.0 提供。

1.2. 关于 MIGRATION TOOLKIT FOR APPLICATIONS

什么是 Migration Toolkit for Applications?

Migration Toolkit for Applications (MTA) 在 Red Hat OpenShift 的混合云环境中加速大规模应用程序现代化的过程。此解决方案会考虑整个迁移过程的详细情况，包括产品组合和应用程序级别的信息：库存、评估、分析和管理应用程序，以便更快地通过用户界面迁移到 OpenShift。

MTA 使用大量默认问题作为评估应用程序的基础，或者您可以创建自己的自定义问题，允许您估算为容器化准备应用程序所需的难度、时间和其他资源。您可以使用评估结果作为与利益相关者进行的讨论基础，以确定哪些应用程序可以被容器化，哪些需要大量的准备工作，哪些不适用于容器化。

MTA 会根据一个或多个规则集来对应用程序进行分析，并找出应用程序的哪些部分需要进行修改才可以对其进行现代化。

MTA 检查应用程序工件，包括项目源目录和应用程序存档，然后生成 HTML 报告突出显示需要更改的区域。

Migration Toolkit for Applications 如何简化迁移？

Migration Toolkit for Applications 会查找常见资源和在迁移应用程序时的已知问题。它为应用程序使用的技术提供了高级视图。

MTA 生成详细的报告，评估迁移或现代化路径。此报告可帮助您估算大型项目所需的工作量，并减少涉及的工作。

1.3. 关于用户界面

借助 Migration Toolkit for Applications 的用户界面，用户可以以团队的方式评估和报错应用程序，以防出现风险并适合迁移到 Red Hat OpenShift 上的混合云环境。

使用用户界面评估和分析应用程序，以便在库存、评估、分析和管理应用程序更快地迁移到 OpenShift 时了解采用过程中的潜在缺陷。

第 2 章 用户界面视图

Migration Toolkit for Applications (MTA) 用户界面有两个视图：

- 管理视图
- 迁移视图

在 管理视图中，您将配置实例环境，使用凭证、存储库、HTTP 和 HTTPS 代理定义、自定义迁移目标和问题管理。

在迁移视图中，您将执行应用程序评估和分析、审核报告并添加用于评估和分析的应用程序。

第 3 章 安装 MIGRATION TOOLKIT FOR APPLICATIONS 用户界面

您可以在所有 Red Hat OpenShift 云服务和 Red Hat OpenShift 自我管理的版本上安装 Migration Toolkit for Applications (MTA)用户界面。

重要

为了能够创建 MTA 实例，您必须首先安装 MTA Operator。

MTA Operator 是一个结构性层，用于管理 OpenShift 上部署的资源，如数据库、前端和后端，以自动创建 MTA 实例。

3.1. 持久性卷要求

要成功部署，MTA Operator 需要 3 个 RWO 持久性卷 (PV)供不同组件使用。如果 **rwx_supported** 配置选项设为 **true**，则 MTA Operator 需要额外的 2 RWX PV，供 Maven 和 hub 文件存储使用。下表中描述了 PV：

表 3.1. 所需的持久性卷

Name	默认大小	访问模式	Description
hub 数据库	10 GiB	RWO	hub 数据库
hub bucket	100 GiB	RWX	Hub 文件存储；如果 **rwx_supported** 配置选项被设置为 **true**，则需要此项
keycloak postgresql	1 GiB	RWO	Keycloak 后端数据库
pathfinder postgresql	1 GiB	RWO	Pathfinder 后端数据库
缓存	100 GiB	RWX	Maven m2 缓存；如果 **rwx_supported** 配置选项设为 **true**，则需要此项

3.2. 安装 MIGRATION TOOLKIT FOR APPLICATIONS OPERATOR 和用户界面

您可以在 Red Hat OpenShift 版本 4.13-4.15 上安装 Migration Toolkit for Applications (MTA)和用户界面。

先决条件

- 4 个 vCPU、8 GiB RAM 和 40 GiB 持久性存储。

- 任何云服务或自托管版本的 Red Hat OpenShift 在版本 4.13-4.15。
- 您必须以具有 **cluster-admin** 权限的用户身份登录。

如需更多信息，请参阅 OpenShift Operator 生命周期。

流程

1. 在 Red Hat OpenShift Web 控制台中，点 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** 字段搜索 **MTA**。
3. 点 **Migration Toolkit for Applications** Operator，然后点 **Install**。
4. 在 **Install Operator** 页面中，点 **Install**。
5. 点 **Operators → Installed Operators** 来验证 MTA Operator 是否出现在 **openshift-mta** 项目中，状态为 **Succeeded**。
6. 点 **MTA** Operator。
7. 在 **Provided APIs** 下，找到 **Tackle**，然后点 **Create Instance**。
   Create Tackle 窗口在 **Form** 视图中打开。
8. 查看自定义资源(CR)设置。默认的选择应该可以被接受，但请确保检查系统对存储、内存和内核的要求。
9. 要直接使用 YAML 文件，请点击 **YAML** 视图，并查看 YAML 文件的 **spec** 部分中列出的 CR 设置。
   这个表中列出了最常用的 CR 设置：

   表 3.2. Tackle CR 设置

Name	default	描述
cache_data_volume_size	**100 GiB**	为缓存卷请求的大小；当 **rwx_supported=false** 时会被忽略
cache_storage_class	默认存储类	用于缓存卷的存储类；当 **rwx_supported=false** 时会被忽略
feature_auth_required	**True**	用于指明是否需要 keycloak 授权（单用户/"noauth"）的标记
feature_isolate_namespace	**True**	指定是否启用使用网络策略进行命名空间隔离的标志
hub_database_volume_size	**10 GiB**	为 Hub 数据库卷请求的大小
hub_bucket_volume_size	**100 GiB**	为 Hub 存储桶卷请求的大小
hub_bucket_storage_class	默认存储类	用于存储桶卷的存储类

Name	default	描述
keycloak_database_data_volume_size	**1 GiB**	为 Keycloak 数据库卷请求的大小
pathfinder_database_data_volume_size	**1 GiB**	为 Pathfinder 数据库卷请求的大小
maven_data_volume_size	**100 GiB**	为 Maven m2 缓存卷请求的大小；在 MTA 6.0.1 中弃用
rwx_storage_class	不适用	为 Tackle RWX 卷请求的存储类；在 MTA 6.0.1 中弃用
rwx_supported	**True**	指明集群存储是否支持 RWX 模式的标记
rwo_storage_class	不适用	为 Tackle RW0 卷请求的存储类
rhsso_external_access	**False**	指明是否创建了专用路由来访问 MTA 受管 RHSSO 实例的标志
analyzer_container_limits_cpu	**1**	pod 允许使用的最大 CPU 数量
analyzer_container_limits_memory	**4GiB**	pod 允许使用的最大内存量。如果 pod 显示 **OOMKilled** 错误，您可以增加这个限制。
analyzer_container_requests_cpu	**1**	pod 需要运行的最少 CPU 数量
analyzer_container_requests_memory	**4GiB**	pod 需要运行的最小内存量

YAML 文件示例

```
kind: Tackle
apiVersion: tackle.konveyor.io/v1alpha1
metadata:
  name: mta
  namespace: openshift-mta
spec:
  hub_bucket_volume_size: "25Gi"
  maven_data_volume_size: "25Gi"
  rwx_supported: "false"
```

10. 如果需要，编辑 CR 设置，然后点 **Create**。
11. 在管理视图中，点 **Workloads → Pods** 来验证 MTA Pod 是否正在运行。
12. 使用 OpenShift 中的 **mta-ui** 应用公开的路由，从浏览器访问用户界面。

13. 使用以下凭证登录：
    - 用户名 ：admin
    - 密码 ： Passw0rd!
14. 出现提示时，创建新密码。

3.3. 在断开连接的 RED HAT OPENSHIFT 环境中安装 MIGRATION TOOLKIT FOR APPLICATIONS OPERATOR

您可以根据 通用流程中 的说明在断开连接的环境中安装 MTA Operator。

在通用流程的第 1 步中，按如下所示为镜像配置镜像集：

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v1alpha2
storageConfig:
  registry:
    imageURL: registry.to.mirror.to
    skipTLS: false
mirror:
  operators:
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15
    packages:
    - name: mta-operator
      channels:
      - name: stable-v7.0
    - name: rhsso-operator
      channels:
      - name: stable
  helm: {}
```

3.4. 在 RED HAT OPENSHIFT LOCAL 上运行 MTA 的内存要求

当在 Red Hat OpenShift Local 上安装时，MTA 需要最少的内存量来完成其分析。添加内存可加快分析过程的运行速度。下表描述了具有不同内存量的 MTA 性能。

表 3.3. OpenShift Local MTA 内存要求

内存(GiB)	描述
10	因为内存不足，MTA 无法运行分析
11	因为内存不足，MTA 无法运行分析
12	**MTA 正常工作，分析将在大约 3 分钟内完成**
15	MTA 正常工作，分析在 2 分钟内完成
20	MTA 可以快速工作，分析在不到 1 分钟内完成

测试结果表示在 OpenShift Local 上运行 MTA 的最小内存量为 **12 GiB**。

注意

- 测试是通过用户界面运行 MTA 二进制文件分析来实现的。
- 所有分析都使用 **tackle-testapp** 二进制文件。
- 所有测试都在 OpenShift Local 集群上执行，没有安装监控工具。
- 安装集群监控工具需要额外的 5 GiB 内存。

3.4.1. 驱除阈值

每个节点分配有一定数量的内存。一些内存是为系统服务保留的。其余内存用于运行 pod。如果 pod 使用超过分配的内存量，则会触发内存不足事件，节点终止并显示 **OOMKilled** 错误。

要防止内存不足事件和保护节点，请使用 **--eviction-hard** 设置。此设置指定节点驱除 pod 的内存可用性阈值。设置的值可以是绝对或百分比。

节点内存分配设置示例

- 节点容量：**32 GiB**
- **--system-reserved** 设置：**3 GiB**
- **--eviction-hard** 设置 ：**100 MiB**

此节点上运行 pod 的内存量为 28.9 GiB。这个数量是通过从节点的总容量中减去 **system-reserved** 和 **eviction-hard** 值来计算的。如果内存用量超过这个数量，节点将开始驱除 pod。

3.5. 红帽单点登录

MTA 将身份验证和授权委托给 MTA operator 管理的 Red Hat Single Sign-On (RHSSO) 实例。除了控制受管 RHSSO 实例的完整生命周期外，MTA operator 还管理一个专用域的配置，其中包含 MTA 所需的所有角色和权限。

如果在 MTA 管理的 RHSSO 实例中需要高级配置，如 为 User Federation 添加一个供应商 或集成身份提供程序，用户可以通过 **mta-ui** 路由中的 **/auth/admin** 子路径登录到 RHSSO 管理控制台。用于访问 MTA 管理的 RHSSO 实例的 admin 凭据可以从安装用户界面的命名空间中提供的 **credential-mta-rhsso** secret 中检索。

通过在管理 MTA 实例的 **Tackle CR** 中将 **rhsso_external_access** 参数设置为 **True** 来创建 MTA 受管 RHSSO 实例的专用路由。

如需更多信息，请参阅 Red Hat Single Sign-On 功能和概念 。

3.5.1. 角色和权限

下表包含 MTA 查找受管 RHSSO 实例的角色和权限（范围）：

tackle-admin	资源名称	Verbs

	附加组件	delete get post put
	adoptionplans	post
	应用程序	delete get post put
	applications.facts	delete get post put
	applications.tags	delete get post put
	applications.bucket	delete get post put
	assessments	delete get patch post put
	businessservices	delete get post put
	dependencies	delete get post put
	identities	delete get post put

	imports	delete get post put
	jobfunctions	delete get post put
	proxies	delete get post put
	reviews	delete get post put
	设置	delete get post put
	stakeholdergroups	delete get post put
	stakeholders	delete get post put
	tags	delete get post put
	tagtypes	delete get post put
	tasks	delete get post put

	tasks.bucket	delete get post put
	tickets	delete get post put
	trackers	delete get post put
	缓存	delete get
	files	delete get post put
	rulebundles	delete get post put
tackle-architect	**资源名称**	**Verbs**
	附加组件	delete get post put
	applications.bucket	delete get post put
	adoptionplans	post
	应用程序	delete get post put
	applications.facts	delete get post put

	applications.tags	delete get post put
	assessments	delete get patch post put
	businessservices	delete get post put
	dependencies	delete get post put
	identities	get
	imports	delete get post put
	jobfunctions	delete get post put
	proxies	get
	reviews	delete get post put
	设置	get
	stakeholdergroups	delete get post put
	stakeholders	delete get post put

	tags	delete get post put
	tagtypes	delete get post put
	tasks	delete get post put
	tasks.bucket	delete get post put
	trackers	get
	tickets	delete get post put
	缓存	get
	files	delete get post put
	rulebundles	delete get post put
tackle-migrator	**资源名称**	**Verbs**
	附加组件	get
	adoptionplans	post
	应用程序	get
	applications.facts	get

	applications.tags	get
	applications.bucket	get
	assessments	get post
	businessservices	get
	dependencies	delete get post put
	identities	get
	imports	get
	jobfunctions	get
	proxies	get
	reviews	get post put
	设置	get
	stakeholdergroups	get
	stakeholders	get
	tags	get
	tagtypes	get
	tasks	delete get post put
	tasks.bucket	delete get post put
	tackers	get

	tickets	get
	缓存	get
	files	get
	rulebundles	get

第 4 章 配置实例环境

您可以在 **Administration** 视图中配置以下内容：

- General
- 凭证
- 软件仓库
- HTTP 和 HTTPS 代理设置
- 自定义迁移目标
- 问题管理

4.1. GENERAL

您可以启用或禁用以下选项：

- 检查应用程序而无需首先运行评估
- 下载 HTML 报告
- 下载 CSV 报告

4.2. 配置凭证

您可以在 **Administration** 视图中配置以下类型的凭证：

- 源控制
- Maven
- Proxy

4.2.1. 配置源控制凭证

您可以在 Migration Toolkit for Applications (MTA) 用户界面的 **Credentials** 视图中配置源控制凭证。

流程

1. 在 **Administration** 视图中，点 **Credentials**。
2. 点 **Create new**。
3. 输入以下信息：
   - Name
   - 描述（可选）
4. 在 **Type** 列表中，选择 **Source Control**。
5. 在 **User credentials** 列表中，选择 **Credential Type** 并输入请求的信息：

- 用户名/密码
  - 用户名
  - 密码（隐藏）
- SCM 私钥/密码
  - SCM 私钥
  - 私钥密码（隐藏）

注意

特定于类型的凭据信息（如密钥和密码短语）是隐藏的，或显示为 [Encrypted]。

6. 点 **Create**。
   MTA 验证输入并创建新凭证。必须解析和检查 SCM 密钥以获取有效性。如果验证失败，则会显示以下错误消息：**"not a valid key/XML file"**。

4.2.2. 配置 Maven 凭证

您可以在 Migration Toolkit for Applications (MTA) 用户界面的 **Credentials** 视图中配置新的 Maven 凭证。

流程

1. 在 **Administration** 视图中，点 **Credentials**。
2. 点 **Create new**。
3. 输入以下信息：
   - Name
   - 描述（可选）
4. 在 **Type** 列表中，选择 **Maven Settings File**。
5. 上传设置文件或粘贴其内容。
6. 点 **Create**。
   MTA 验证输入并创建新凭证。必须解析 Maven **settings.xml** 文件，并检查其有效。如果验证失败，则会显示以下错误消息：**"not a valid key/XML file"**。

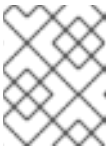
4.2.3. 配置代理凭证

您可以在 Migration Toolkit for Applications (MTA) 用户界面的 **Credentials** 视图中配置代理凭证。

流程

1. 在 **Administration** 视图中，点 **Credentials**。
2. 点 **Create new**。

3. 输入以下信息：
   - Name
   - 描述（可选）
4. 在 **Type** 列表中，选择 **Proxy**。
5. 输入以下信息。
   - 用户名
   - 密码

   注意

   特定于类型的凭据信息（如密钥和密码短语）是隐藏的，或显示为 [Encrypted]。
6. 点 **Create**。
   MTA 验证输入并创建新凭证。

4.3. 配置软件仓库

您可以在 **Administration** 视图中配置以下类型的软件仓库：

- Git
- Subversion
- Maven

4.3.1. 配置 Git 存储库

您可以在 Migration Toolkit for Applications (MTA) 用户界面的 **Repositories** 视图中配置 Git 存储库。

流程

1. 在 **Administration** 视图中，点 **Repositories**，然后点 **Git**。
2. 将 **Consume insecure Git** 存储库开关切换到右侧。

4.3.2. 配置子版本软件仓库

您可以在 Migration Toolkit for Applications (MTA) 用户界面的 **Repositories** 视图中配置 subversion 软件仓库。

流程

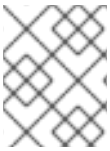
1. 在 **Administration** 视图中，点 **Repositories**，然后点 **Subversion**。
2. 将 **Consume insecure Subversion** 软件仓库切换到右边。

4.3.3. 配置 Maven 存储库并缩小其大小

您可以使用 MTA 用户界面配置 Maven 存储库并缩小其大小。

4.3.3.1. 配置 Maven 存储库

您可以在 Migration Toolkit for Applications (MTA) 用户界面的 **Repositories** 视图中配置 Maven 存储库。



注意

如果 Tackle CR 的 **rwx_supported** 配置选项被设置为 **false**，则 **Consume insecure artifact repositories** 开关会被禁用，且无法进行这个过程。

流程

1. 在 **Administration** 视图中，点 **Repositories**，然后点 **Maven**。
2. 将 **Consume insecure artifact** 存储库切换到右边。

4.3.3.2. 减少 Maven 存储库的大小

您可以在 Migration Toolkit for Applications (MTA) 用户界面的 **Repositories** 视图中减小 Maven 存储库的大小。

注意

如果 Tackle CR 的 **rwx_supported** 配置选项被设置为 **false**，则 **Local artifact repository** 字段和 **Clear repository** 按钮都被禁用，且无法执行此步骤。

流程

1. 在 **Administration** 视图中，点 **Repositories**，然后点 **Maven**。
2. 点 **Clear repository** 链接。

注意

取决于存储库的大小，即使功能正常工作，大小变化可能无法识别。

4.4. 配置 HTTP 和 HTTPS 代理设置

您可以使用此管理模块配置 HTTP 和 HTTPS 代理设置。

流程

1. 在 **Administration** 视图中，点 **Proxy**。
2. 使用 **HTTP 代理**或 **HTTPS 代理**以启用代理连接。
3. 输入以下信息：
   - 代理主机
   - 代理端口

4. 可选：切换 **HTTP 代理凭证** 或 **HTTPS 代理凭证**来启用身份验证。

5. 点 **Insert**。

4.5. SEEDING 一个实例

如果您是项目架构师，可以在迁移前在 Controls 窗口中配置实例的关键参数。可以根据需要添加和删除参数。以下参数定义了在一个机构中会受迁移影响或参阅迁移的应用程序、个人、团队、以及在机构中的不同部门：

- **利益相关者**
- stakeholder 组
- **任务功能**
- 业务服务
- 标签类别
- Tags

您可以以任何顺序创建和配置实例。但是，以下建议的顺序是创建**利益相关者**和标签最有效的。

利益相关者：

1. 创建 Stakeholder 组
2. 创建**作业功能**
3. 创建 Stakeholders

标签：

1. 创建标签类别
2. 创建标签

利益相关者，并通过以下方法定义：

- 电子邮件
- Name
- **作业功能**
- stakeholder 组

4.5.1. 创建新的拥有者组

没有定义默认的拥有者组。您可以按照以下步骤创建新的拥有者组。

流程

1. 在 **Migration** 视图中，点 **Controls**。

2. 点 **Stakeholder groups**。
3. 点 **Create new**。
4. 输入以下信息：
   - Name
   - 描述
   - 成员
5. 点 **Create**。

4.5.2. 创建新作业功能

Migration Toolkit for Applications (MTA) 使用 job function 属性来分类利益相关者，并提供可扩展的默认值列表。

您可以按照以下步骤创建新作业功能，使其不在默认列表中。

流程

1. 在 **Migration** 视图中，点 **Controls**。
2. 点 **Job functions**。
3. 点 **Create new**。
4. 在 **Name** 文本框中输入作业功能标题。
5. 点 **Create**。

4.5.3. 创建新的拥有者

您可以按照以下步骤创建新的迁移项目拥有者。

流程

1. 在 **Migration** 视图中，点 **Controls**。
2. 点 **Stakeholders**。
3. 点 **Create new**。
4. 输入以下信息：
   - 电子邮件
   - Name
   - 作业功能 - 可以创建自定义功能
   - stakeholder 组
5. 点 **Create**。

4.5.4. 创建新业务服务

Migration Toolkit for Applications (MTA)使用业务服务属性来指定使用应用程序以及迁移影响的部门。

您可以按照以下过程创建新业务服务。

流程

1. 在 **Migration** 视图中，点 **Controls**。
2. 点 **Business services**。
3. 点 **Create new**。
4. 输入以下信息：
   - Name
   - 描述
   - 所有者
5. 点 **Create**。

4.5.5. 创建新标签类别

Migration Toolkit for Applications (MTA) 使用多个类别中的标签，并提供默认值列表。您可以按照以下步骤创建新标签类别。

流程

1. 在 **Migration** 视图中，点 **Controls**。
2. 点 **Tags**。
3. 点 **Create tag category**。
4. 输入以下信息：
   - Name
   - Rank - 标签出现在应用程序中的顺序
   - Color
5. 点 **Create**。

4.5.5.1. 创建新标签

您可以按照以下步骤创建新标签，该标签不在默认列表中。

流程

1. 在 **Migration** 视图中，点 **Controls**。
2. 点 **Tags**。

3. 点 **Create tag**。
4. 输入以下信息：
    - Name
    - 标签类别
5. 点 **Create**。

第 5 章 创建和配置 JIRA 连接

您可以通过从 MTA 用户界面为每个迁移创建一个 JIRA 问题来跟踪应用程序迁移。要能够创建 JIRA 问题，您首先需要执行以下操作：

1. 创建一个 MTA 凭证，以向您在上一步中创建的 JIRA 实例的 API 进行身份验证。
2. 在 MTA 中创建 JIRA 实例并建立与该实例的连接。

5.1. 配置 JIRA 凭证

要在 MTA 中定义 JIRA 实例并建立与该实例的连接，您必须首先创建一个 MTA 凭证来向 JIRA 实例的 API 进行身份验证。

有两种类型的凭证可用：

- 基本的 **auth** - 用于 JIRA 云和私有 JIRA 服务器或数据中心
- **bearer Token** - 用于私有 JIRA 服务器或数据中心

要创建 MTA 凭证，请按照以下步骤操作。

流程

1. 在 **Administration** 视图中，点 **Credentials**。
   Credentials 页面将打开。
2. 点 **Create new**。
3. 输入以下信息：
   - **Name**
   - 描述 （可选）
4. 在 **Type** 列表中，选择 **Basic Auth (JIRA)** 或 **Bearer Token (JIRA)** ：
   - 如果您选择了 **Basic Auth (JIRA)**，如下所示：
     a. 在 **Email** 字段中，输入您的电子邮件。
     b. 在 **Token** 字段中，根据特定的 JIRA 配置，输入 JIRA 站点或 JIRA 登录密码上生成的令牌。

     注意

     要获取 JIRA 令牌，您需要登录到 JIRA 站点。

     c. 点击 **Save**。
     新凭证会出现在 **Credentials** 页面中。
   - 如果您选择了 **Bearer Token (JIRA)**，如下所示：
     a. 在 **Token** 字段中输入 JIRA 站点上生成的令牌。
     b. 点击 **Save**。

新凭证会出现在 **Credentials** 页面中。

您可以点击 **Edit** 来编辑凭证。

若要删除凭据，请单击 **Delete**。

注意

您不能删除已分配给 JIRA 连接实例的凭证。

5.2. 创建和配置 JIRA 连接

要在 MTA 中创建 JIRA 实例并建立与该实例的连接，请按照以下步骤操作。

流程

1. 在 **Administration** 视图中，在 **Issue Management** 下点 **JIRA**。
   此时会打开 **JIRA** 配置页面。
2. 点 **Create new**。
   此时会打开 **New instance** 窗口。
3. 输入以下信息：
   - 实例的名称
   - JIRA 帐户的 Web 界面的 URL
   - 实例类型 - 从列表中选择 **JIRA Cloud** 或 **Jira Server/Data** Center
   - credentials - 从列表中选择

   注意

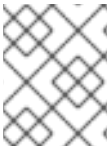
   如果所选实例类型是 **JIRA Cloud**，则列表中仅显示 **Basic Auth** 凭证。

   如果所选的实例类型是 **JIRA Server/Data center**，则会显示 **Basic Auth** 和 **Token Bearer** 凭证。选择适合您的 JIRA 服务器或数据中心的特定配置的类型。
4. 默认情况下，无法使用无效证书的服务器建立连接。要覆盖此限制，请切换 **Enable insecure communication** switch。
5. 点 **Create**。
   新的连接实例会出现在 **JIRA** 配置 页面中。

   连接建立并授权后，连接 列中的状态将变为 **Connected**。

   如果 **Connection** 状态变为 **Not connected**，请单击状态以查看错误的原因。

JIRA 配置 表根据 名称和 **URL** 过滤，并根据 实例名称 和 **URL** 排序。



注意

只要 JIRA 中存在问题，用于为迁移 wave 创建问题的 JIRA 连接不能被删除，即使迁移的 wave 被删除。

第 6 章 使用 MTA 管理应用程序

您可以使用 Migration Toolkit for Applications (MTA)用户界面执行以下任务：

- 添加应用程序。
- 分配应用凭据。
- 导入应用程序列表。
- 下载用于导入应用程序的 CSV 模板。
- 创建应用程序迁移。
- 为迁移 waves 创建 JIRA 问题。

MTA 用户界面应用程序有以下属性：

- 名称（自由文本）
- 描述（可选，自由文本）
- 业务服务（可选，从列表中选择）
- 标签（可选，从列表中选择）
- 所有者（可选，从列表中选择）
- 贡献者（可选，从列表中选择）
- 源代码（用户输入的路径）
- 二进制（用户输入的路径）

6.1. 添加新应用程序

您可以将新应用程序添加到应用程序清单中，以进行后续评估和分析。

提示

在创建应用程序之前，设置业务服务，检查标签和标签类别，并根据需要创建添加。

先决条件

- 您已登录到 MTA 服务器。

流程

1. 在 Migration 视图中，点 Application Inventory。
2. 点 Create new。
3. 在 Basic information 下，输入以下字段：
    - 名称 ：新应用的唯一名称。

- 描述 ：应用程序的简短描述（可选）。
- 业务服务 ：应用程序的目的（可选）。
- 手动标记 ：代表应用程序的软件标签（可选，一个或多个）。
- 所有者 ：从下拉列表中选择注册的软件所有者（可选）。
- 供稿人 ：从下拉列表中选择 Contributors （可选，一个或多个）。
- 注释 ：应用程序上的相关注释（可选）。

4. 点 Source Code 并输入以下字段：
   - 存储库类型 ：Git 或 Subversion。
   - 源存储库 ：保存软件代码的存储库的 URL。
   - 分支 ：存储库中的应用程序代码分支（可选）。
   - 根路径 ：目标应用程序的仓库中的 root 路径（可选）。

   注意：如果您在 Branch 或 Root 路径字段中输入任何值，则 Source repository 字段将变为必需。
5. 可选：点 Binary 并输入以下字段：
   - 组 ：应用程序工件的 Maven 组。
   - 工件 ：应用程序的 Maven 工件。
   - 版本 ：应用程序的软件版本。
   - 打包 ：应用程序工件的打包，如 JAR、WAR 或 EAR。

   注意： 如果您在任何 Binary 部分字段中输入任何值，则所有字段都会自动成为强制的。
6. 点 Create。新应用程序会出现在定义的应用程序列表中。

6.2. 编辑应用程序

您可以编辑应用程序清单中的现有应用程序，并重新运行此应用程序的评估或分析。

先决条件

- 您已登录到 MTA 服务器。

流程

1. 在 Migration 视图中，点 Application Inventory。
2. 选择 Migration working 模式。
3. 单击左侧菜单栏中的 Application Inventory。可用应用程序列表会出现在主窗格中。
4. 点 Edit ( )打开应用程序设置。

5. 检查应用设置。有关应用程序设置列表，请参阅 添加应用程序。
6. 如果您更改了任何应用程序设置，请单击 **Save**。

6.3. 为应用程序分配凭证

您可以为一个或多个应用程序分配凭证。

流程

1. 在 **Migration** 视图中，点 **Application inventory**。
2. 点击 **analyzee** 右侧的 **Options** 菜单( )，然后选择 **Manage credentials**。
3. 从 **Source credentials** 列表中选择一个凭据，再从 **Maven** 设置列表中选择一个 凭据。
4. 点 **Save**。

6.4. 导入应用程序列表

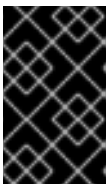
您可以将包含应用程序及其属性列表的 **.csv** 文件导入到 **Migration Toolkit for Applications (MTA)**用户界面。

注意

导入应用程序列表不会覆盖任何现有的应用程序。

流程

1. 检查导入文件，以确保它包含所有必需的信息（以所需格式表示）。
2. 在 **Migration** 视图中，点 **Application Inventory**。
3. 点击 **Options** 菜单( )。
4. 点 **Import**。
5. 选择所需的文件，然后单击 **Open**。
6. 可选： 选择 启用自动创建缺少的实体。默认选择这个选项。
7. 验证导入已完成，并检查接受或拒绝的行数。
8. 点复选框左侧的箭头来查看导入的应用程序。



重要

接受的行可能与应用程序清单列表中的应用程序数量不匹配，因为某些行是依赖项。要验证，请检查 **CSV** 文件的 **Record Type** 列，以了解定义为 **1** 的应用程序以及定义为 **2** 的依赖项。

6.5. 下载 CSV 模板

您可以使用 Migration Toolkit for Applications (MTA)用户界面下载用于导入应用程序列表的 CSV 模板。

流程

1. 在 Migration 视图中，点 Application inventory。
2. 点击 Review 右侧的 Options 菜单( )。
3. 点 Manage import 以打开 Application import 页面。
4. 点击 Import 右侧的 Options 菜单( )。
5. 点 Download CSV 模板。

6.6. 创建迁移 WAVE

迁移 wave 是一个组应用程序，您可以根据给定时间表迁移。您可以通过将 wave 的应用列表导出到 JIRA 问题管理系统来跟踪每个迁移。这会自动为迁移 wave 的每个应用程序创建一个单独的 JIRA 问题。

流程

1. 在 Migration 视图中，点 Migration waves。
2. 点 Create new。此时会打开 New migration wave 窗口。
3. 输入以下信息：
   - name （可选）。如果没有指定名称，您可以使用开始和结束日期来识别迁移 waves。
   - 潜在的开始日期.此日期必须早于当前日期。
   - 潜在结束日期.此日期必须早于开始日期。
   - 利益相关者 （可选）
   - stakeholder 组 （可选）
4. 点 Create。新的迁移 wave 会出现在现有迁移 Waves 列表中。
5. 要将应用程序分配给迁移 wave，点迁移wave 右侧的 Options 菜单( )，然后选择 Manage applications。
   此时会打开 Manage applications 窗口，显示没有分配给任何其他迁移 Wave 的应用程序列表。
6. 选择您要分配给迁移的应用程序的复选框。
7. 点 Save。

注意

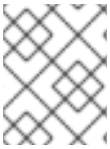
与迁移 wave 关联的每个应用程序的所有者和贡献者会自动添加到迁移利益相关者列表中。

8. 可选： 要更新迁移 wave，请从迁移 wave 的 Options 菜单选择 Update ( )。此时会打开 Update migration wave 窗口。

6.7. 为迁移 WAVE 创建 JIRA 问题

您可以使用迁移 wave 为分配给迁移的每个应用程序自动创建 JIRA 问题。为与迁移 wave 关联的每个应用程序创建一个单独的 JIRA 问题。每个问题的以下字段都会被自动填写：

- title: **Migrate <application name>**
- reporter ：令牌所有者的 Username。
- 描述 ： 由 **Konveyor**创建



注意

如果应用程序链接到 JIRA ticket 或与迁移 wave 关联，则无法删除应用程序。要从 JIRA 票据中取消链接应用程序，请点击应用程序的详情视图中的 **JIRA** 图标的 **Unlink**。

先决条件

- 您已配置了 JIRA 连接。如需更多信息，请参阅 创建和配置 JIRA 连接。

流程

1. 在 Migration 视图中，点 Migration waves。
2. 点击您要为其创建 JIRA 的迁移wave 右侧的 Options 菜单( )，然后选择 Export to Issue Manager。此时会打开 Export to Issue Manager 窗口。
3. 选择 JIRA Cloud 或 Jira Server/Datacenter 实例类型。
4. 从列表中选择 instance、project 和 issue 类型。
5. 单击 Export。Migration waves 页面的迁移状态更改为 创建的问题。
6. 可选： 要查看迁移的每个应用程序的状态，请点 Status 列。
7. 可选： 要查看任何特定应用程序是否与迁移 wave 关联，请在 Application inventory 页面中打开应用程序的 Details 选项卡。

第 7 章 使用 MTA 评估和分析应用程序

您可以使用 Migration Toolkit for Applications (MTA)用户界面来评估和分析应用程序：

- 在评估应用程序时，MTA 会估算准备应用程序进行容器化的风险和成本，包括时间、人员和其他因素。您可以使用评估结果来讨论利益相关者之间的讨论，以确定应用程序是否适合容器化。
- 分析应用程序时，MTA 使用规则来确定应用程序中的哪些特定行，然后才能迁移或现代化应用程序。

7.1. ASSESSMENT 模块功能

Migration Toolkit for Applications (MTA) 评估 模块提供以下功能来评估和分析应用程序：

评估中心

评估 中心与应用清单 集成。

增强的评估问题功能

在 MTA 7.0 中，您可以导入和导出评估问题。您还可以使用 YAML 语法设计带有可下载模板的自定义问题，其中包括以下功能：

- 条件问题：如果此应用程序或架构中存在特定标签，您可以根据应用程序或架构类型包含或排除问题。
- 基于答案的应用程序自动标记：如果提供了某个回答，您可以定义要应用到应用程序或架构的标签。
- 从应用程序或架构类型中的标签自动回答。

如需更多信息，请参阅自定义 评估问题。

注意

您可以自定义并保存默认问题。如需更多信息，请参阅 默认评估问题。

多个评估问题

评估 模块支持多个问题，与一个或多个应用程序相关。

archetypes

您可以将具有类似特征的应用程序分组到 archetypes 中。这可让您一次性评估多个应用程序。每个架构类型都有共享的税款、利益相关者和利益相关者组。所有应用程序都会继承评估并从其分配的 archetypes 中进行检查。
如需更多信息，请参阅使用 archetypes。

7.2. MTA 评估问题

Migration Toolkit for Applications (MTA)使用评估问题(默认) 或自定义 （默认）来评估应用程序容器化所带来的风险。

评估报告提供有关迁移相关的应用程序和风险的信息。该报告还通过为评估提交的应用程序的优先级、业务关键性和依赖项生成采用计划。

7.2.1. 默认评估问题

传统路径finder 是应用程序的默认 Migration Toolkit (MTA)问题naire。Pathfinder 是一个基于问题的工具，可用于评估应用程序在企业级 Kubernetes 平台上进行现代化的应用程序是否可行。

通过与默认问卷和审查流程交互，系统增强了通过一系列评估报告呈现的应用程序知识。

您可以将默认问题导出到 YAML 文件中：

例 7.1. Legacy Pathfinder YAML 文件

```
name: Legacy Pathfinder
description: ''
sections:
  - order: 1
    name: Application details
    questions:
      - order: 1
        text: >-
          Does the application development team understand and actively develop
          the application?
        explanation: >-
          How much knowledge does the team have about the application's
          development or usage?
        answers:
          - order: 2
            text: >-
              Maintenance mode, no SME knowledge or adequate documentation
              available
            risk: red
            rationale: ''
            mitigation: ''
          - order: 0
            text: unknown
            risk: unknown
            rationale: ''
            mitigation: ''
          - order: 1
            text: >-
              Little knowledge, no development (example: third-party or
              commercial off-the-shelf application)
            risk: red
            rationale: ''
            mitigation: ''
          - order: 3
            text: Maintenance mode, SME knowledge is available
            risk: yellow
            rationale: ''
            mitigation: ''
          - order: 4
            text: Actively developed, SME knowledge is available
            risk: green
            rationale: ''
            mitigation: ''
          - order: 5
            text: greenfield application
```

```
      risk: green
      rationale: ''
      mitigation: ''
  - order: 2
    text: How is the application supported in production?
    explanation: >-
      Does the team have sufficient knowledge to support the application in
      production?
    answers:
      - order: 3
        text: >-
          Multiple teams provide support using an established escalation
          model
        risk: yellow
        rationale: ''
        mitigation: ''
      - order: 0
        text: unknown
        risk: unknown
        rationale: ''
        mitigation: ''
      - order: 1
        text: >-
          External support provider with a ticket-driven escalation process;
          no inhouse support resources
        risk: red
        rationale: ''
        mitigation: ''
      - order: 2
        text: >-
          Separate internal support team, separate from the development
          team, with little interaction between the teams
        risk: red
        rationale: ''
        mitigation: ''
      - order: 4
        text: >-
          SRE (Site Reliability Engineering) approach with a knowledgeable
          and experienced operations team
        risk: green
        rationale: ''
        mitigation: ''
      - order: 5
        text: >-
          DevOps approach with the same team building the application and
          supporting it in production
        risk: green
        rationale: ''
        mitigation: ''
  - order: 3
    text: >-
      How much time passes from when code is committed until the application
      is deployed to production?
    explanation: What is the development latency?
    answers:
      - order: 3
```

```
      text: 2-6 months
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 0
      text: unknown
      risk: unknown
      rationale: ''
      mitigation: ''
    - order: 1
      text: Not tracked
      risk: red
      rationale: ''
      mitigation: ''
    - order: 2
      text: More than 6 months
      risk: red
      rationale: ''
      mitigation: ''
    - order: 4
      text: 8-30 days
      risk: green
      rationale: ''
      mitigation: ''
    - order: 5
      text: 1-7 days
      risk: green
      rationale: ''
      mitigation: ''
    - order: 6
      text: Less than 1 day
      risk: green
      rationale: ''
      mitigation: ''
- order: 4
  text: How often is the application deployed to production?
  explanation: Deployment frequency
  answers:
    - order: 3
      text: Between once a month and once every 6 months
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 0
      text: unknown
      risk: unknown
      rationale: ''
      mitigation: ''
    - order: 1
      text: Not tracked
      risk: red
      rationale: ''
      mitigation: ''
    - order: 2
      text: Less than once every 6 months
      risk: red
```

```
      rationale: ''
      mitigation: ''
    - order: 4
      text: Weekly
      risk: green
      rationale: ''
      mitigation: ''
    - order: 5
      text: Daily
      risk: green
      rationale: ''
      mitigation: ''
    - order: 6
      text: Several times a day
      risk: green
      rationale: ''
      mitigation: ''
- order: 5
  text: >-
    What is the application's mean time to recover (MTTR) from failure in
    a production environment?
  explanation: Average time for the application to recover from failure
  answers:
    - order: 5
      text: Less than 1 hour
      risk: green
      rationale: ''
      mitigation: ''
    - order: 0
      text: unknown
      risk: unknown
      rationale: ''
      mitigation: ''
    - order: 1
      text: Not tracked
      risk: red
      rationale: ''
      mitigation: ''
    - order: 3
      text: 1-7 days
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 2
      text: 1 month or more
      risk: red
      rationale: ''
      mitigation: ''
    - order: 4
      text: 1-24 hours
      risk: green
      rationale: ''
      mitigation: ''
- order: 6
  text: Does the application have legal and/or licensing requirements?
  explanation: >-
```

```
    Legal and licensing requirements must be assessed to determine their
    possible impact (cost, fault reporting) on the container platform
    hosting the application. Examples of legal requirements: isolated
    clusters, certifications, compliance with the Payment Card Industry
    Data Security Standard or the Health Insurance Portability and
    Accountability Act. Examples of licensing requirements: per server,
    per CPU.
  answers:
  - order: 1
    text: Multiple legal and licensing requirements
    risk: red
    rationale: ''
    mitigation: ''
  - order: 0
    text: unknown
    risk: unknown
    rationale: ''
    mitigation: ''
  - order: 2
    text: 'Licensing requirements (examples: per server, per CPU)'
    risk: red
    rationale: ''
    mitigation: ''
  - order: 3
    text: >-
      Legal requirements (examples: cluster isolation, hardware, PCI or
      HIPAA compliance)
    risk: yellow
    rationale: ''
    mitigation: ''
  - order: 4
    text: None
    risk: green
    rationale: ''
    mitigation: ''
- order: 7
  text: Which model best describes the application architecture?
  explanation: Describe the application architecture in simple terms.
  answers:
  - order: 3
    text: >-
      Complex monolith, strict runtime dependency startup order,
      non-resilient architecture
    risk: yellow
    rationale: ''
    mitigation: ''
  - order: 0
    text: unknown
    risk: unknown
    rationale: ''
    mitigation: ''
  - order: 5
    text: Independently deployable components
    risk: green
    rationale: ''
    mitigation: ''
```

```
      - order: 1
        text: >-
          Massive monolith (high memory and CPU usage), singleton
          deployment, vertical scale only
        risk: yellow
        rationale: ''
        mitigation: ''
      - order: 2
        text: >-
          Massive monolith (high memory and CPU usage), non-singleton
          deployment, complex to scale horizontally
        risk: yellow
        rationale: ''
        mitigation: ''
      - order: 4
        text: 'Resilient monolith (examples: retries, circuit breakers)'
        risk: green
        rationale: ''
        mitigation: ''
- order: 2
  name: Application dependencies
  questions:
    - order: 1
      text: Does the application require specific hardware?
      explanation: >-
        OpenShift Container Platform runs only on x86, IBM Power, or IBM Z
        systems
      answers:
        - order: 3
          text: 'Requires specific computer hardware (examples: GPUs, RAM, HDDs)'
          risk: yellow
          rationale: ''
          mitigation: ''
        - order: 0
          text: unknown
          risk: unknown
          rationale: ''
          mitigation: ''
        - order: 1
          text: Requires CPU that is not supported by red Hat
          risk: red
          rationale: ''
          mitigation: ''
        - order: 2
          text: 'Requires custom or legacy hardware (example: USB device)'
          risk: red
          rationale: ''
          mitigation: ''
        - order: 4
          text: Requires CPU that is supported by red Hat
          risk: green
          rationale: ''
          mitigation: ''
    - order: 2
      text: What operating system does the application require?
      explanation: >-
```

```
    Only Linux and certain Microsoft Windows versions are supported in
    containers. Check the latest versions and requirements.
  answers:
    - order: 4
      text: Microsoft Windows
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 0
      text: unknown
      risk: unknown
      rationale: ''
      mitigation: ''
    - order: 1
      text: >-
        Operating system that is not compatible with OpenShift Container
        Platform (examples: OS X, AIX, Unix, Solaris)
      risk: red
      rationale: ''
      mitigation: ''
    - order: 2
      text: Linux with custom kernel drivers or a specific kernel version
      risk: red
      rationale: ''
      mitigation: ''
    - order: 3
      text: 'Linux with custom capabilities (examples: seccomp, root access)'
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 5
      text: Standard Linux distribution
      risk: green
      rationale: ''
      mitigation: ''
- order: 3
  text: >-
    Does the vendor provide support for a third-party component running in
    a container?
  explanation: Will the vendor support a component if you run it in a container?
  answers:
    - order: 2
      text: No vendor support for containers
      risk: red
      rationale: ''
      mitigation: ''
    - order: 0
      text: unknown
      risk: unknown
      rationale: ''
      mitigation: ''
    - order: 1
      text: Not recommended to run the component in a container
      risk: red
      rationale: ''
      mitigation: ''
```

```
    - order: 3
      text: >-
        Vendor supports containers but with limitations (examples:
        functionality is restricted, component has not been tested)
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 4
      text: >-
        Vendor supports their application running in containers but you
        must build your own images
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 5
      text: Vendor fully supports containers, provides certified images
      risk: green
      rationale: ''
      mitigation: ''
    - order: 6
      text: No third-party components required
      risk: green
      rationale: ''
      mitigation: ''
- order: 4
  text: Incoming/northbound dependencies
  explanation: Systems or applications that call the application
  answers:
    - order: 3
      text: >-
        Many dependencies exist, can be changed because the systems are
        internally managed
      risk: green
      rationale: ''
      mitigation: ''
    - order: 0
      text: unknown
      risk: unknown
      rationale: ''
      mitigation: ''
    - order: 4
      text: Internal dependencies only
      risk: green
      rationale: ''
      mitigation: ''
    - order: 1
      text: >-
        Dependencies are difficult or expensive to change because they are
        legacy or third-party
      risk: red
      rationale: ''
      mitigation: ''
    - order: 2
      text: >-
        Many dependencies exist, can be changed but the process is
        expensive and time-consuming
```

```
        risk: yellow
        rationale: ''
        mitigation: ''
      - order: 5
        text: No incoming/northbound dependencies
        risk: green
        rationale: ''
        mitigation: ''
  - order: 5
    text: Outgoing/southbound dependencies
    explanation: Systems or applications that the application calls
    answers:
      - order: 3
        text: Application not ready until dependencies are verified available
        risk: yellow
        rationale: ''
        mitigation: ''
      - order: 0
        text: unknown
        risk: unknown
        rationale: ''
        mitigation: ''
      - order: 1
        text: >-
          Dependency availability only verified when application is
          processing traffic
        risk: red
        rationale: ''
        mitigation: ''
      - order: 2
        text: Dependencies require a complex and strict startup order
        risk: yellow
        rationale: ''
        mitigation: ''
      - order: 4
        text: Limited processing available if dependencies are unavailable
        risk: green
        rationale: ''
        mitigation: ''
      - order: 5
        text: No outgoing/southbound dependencies
        risk: green
        rationale: ''
        mitigation: ''
- order: 3
  name: Application architecture
  questions:
    - order: 1
      text: >-
        How resilient is the application? How well does it recover from
        outages and restarts?
      explanation: >-
        If the application or one of its dependencies fails, how does the
        application recover from failure? Is manual intervention required?
      answers:
        - order: 0
```

```
      text: unknown
      risk: unknown
      rationale: ''
      mitigation: ''
    - order: 1
      text: >-
        Application cannot be restarted cleanly after failure, requires
        manual intervention
      risk: red
      rationale: ''
      mitigation: ''
    - order: 2
      text: >-
        Application fails when a soutbound dependency is unavailable and
        does not recover automatically
      risk: red
      rationale: ''
      mitigation: ''
    - order: 3
      text: >-
        Application functionality is limited when a dependency is
        unavailable but recovers when the dependency is available
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 4
      text: >-
        Application employs resilient architecture patterns (examples:
        circuit breakers, retry mechanisms)
      risk: green
      rationale: ''
      mitigation: ''
    - order: 5
      text: >-
        Application containers are randomly terminated to test resiliency;
        chaos engineering principles are followed
      risk: green
      rationale: ''
      mitigation: ''
- order: 2
  text: How does the external world communicate with the application?
  explanation: >-
    What protocols do external clients use to communicate with the
    application?
  answers:
    - order: 0
      text: unknown
      risk: unknown
      rationale: ''
      mitigation: ''
    - order: 1
      text: 'Non-TCP/IP protocols (examples: serial, IPX, AppleTalk)'
      risk: red
      rationale: ''
      mitigation: ''
    - order: 2
```

```
    text: TCP/IP, with host name or IP address encapsulated in the payload
    risk: red
    rationale: ''
    mitigation: ''
  - order: 3
    text: 'TCP/UDP without host addressing (example: SSH)'
    risk: yellow
    rationale: ''
    mitigation: ''
  - order: 4
    text: TCP/UDP encapsulated, using TLS with SNI header
    risk: green
    rationale: ''
    mitigation: ''
  - order: 5
    text: HTTP/HTTPS
    risk: green
    rationale: ''
    mitigation: ''
- order: 3
  text: How does the application manage its internal state?
  explanation: >-
    If the application must manage or retain an internal state, how is
    this done?
  answers:
  - order: 0
    text: unknown
    risk: unknown
    rationale: ''
    mitigation: ''
  - order: 3
    text: State maintained in non-shared, non-ephemeral storage
    risk: yellow
    rationale: ''
    mitigation: ''
  - order: 1
    text: Application components use shared memory within a pod
    risk: yellow
    rationale: ''
    mitigation: ''
  - order: 2
    text: >-
      State is managed externally by another product (examples:
      Zookeeper or red Hat Data Grid)
    risk: yellow
    rationale: ''
    mitigation: ''
  - order: 4
    text: Disk shared between application instances
    risk: green
    rationale: ''
    mitigation: ''
  - order: 5
    text: Stateless or ephemeral container storage
    risk: green
    rationale: ''
```

```
      mitigation: ''
  - order: 4
    text: How does the application handle service discovery?
    explanation: How does the application discover services?
    answers:
      - order: 0
        text: unknown
        risk: unknown
        rationale: ''
        mitigation: ''
      - order: 1
        text: >-
          Uses technologies that are not compatible with Kubernetes
          (examples: hardcoded IP addresses, custom cluster manager)
        risk: red
        rationale: ''
        mitigation: ''
      - order: 2
        text: >-
          Requires an application or cluster restart to discover new service
          instances
        risk: red
        rationale: ''
        mitigation: ''
      - order: 3
        text: >-
          Uses technologies that are compatible with Kubernetes but require
          specific libraries or services (examples: HashiCorp Consul,
          Netflix Eureka)
        risk: yellow
        rationale: ''
        mitigation: ''
      - order: 4
        text: Uses Kubernetes DNS name resolution
        risk: green
        rationale: ''
        mitigation: ''
      - order: 5
        text: Does not require service discovery
        risk: green
        rationale: ''
        mitigation: ''
  - order: 5
    text: How is the application clustering managed?
    explanation: >-
      Does the application require clusters? If so, how is clustering
      managed?
    answers:
      - order: 0
        text: unknown
        risk: unknown
        rationale: ''
        mitigation: ''
      - order: 1
        text: 'Manually configured clustering (example: static clusters)'
        risk: red
```

```
      rationale: ''
      mitigation: ''
    - order: 2
      text: Managed by an external off-PaaS cluster manager
      risk: red
      rationale: ''
      mitigation: ''
    - order: 3
      text: >-
        Managed by an application runtime that is compatible with
        Kubernetes
      risk: green
      rationale: ''
      mitigation: ''
    - order: 4
      text: No cluster management required
      risk: green
      rationale: ''
      mitigation: ''
- order: 4
  name: Application observability
  questions:
    - order: 1
      text: How does the application use logging and how are the logs accessed?
      explanation: How the application logs are accessed
      answers:
        - order: 0
          text: unknown
          risk: unknown
          rationale: ''
          mitigation: ''
        - order: 1
          text: Logs are unavailable or are internal with no way to export them
          risk: red
          rationale: ''
          mitigation: ''
        - order: 2
          text: >-
            Logs are in a custom binary format, exposed with non-standard
            protocols
          risk: red
          rationale: ''
          mitigation: ''
        - order: 3
          text: Logs are exposed using syslog
          risk: yellow
          rationale: ''
          mitigation: ''
        - order: 4
          text: Logs are written to a file system, sometimes as multiple files
          risk: yellow
          rationale: ''
          mitigation: ''
        - order: 5
          text: 'Logs are forwarded to an external logging system (example: Splunk)'
          risk: green
```

```
      rationale: ''
      mitigation: ''
    - order: 6
      text: 'Logs are configurable (example: can be sent to stdout)'
      risk: green
      rationale: ''
      mitigation: ''
- order: 2
  text: Does the application provide metrics?
  explanation: >-
    Are application metrics available, if necessary (example: OpenShift
    Container Platform collects CPU and memory metrics)?
  answers:
    - order: 0
      text: unknown
      risk: unknown
      rationale: ''
      mitigation: ''
    - order: 1
      text: No metrics available
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 2
      text: Metrics collected but not exposed externally
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 3
      text: 'Metrics exposed using binary protocols (examples: SNMP, JMX)'
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 4
      text: >-
        Metrics exposed using a third-party solution (examples: Dynatrace,
        AppDynamics)
      risk: green
      rationale: ''
      mitigation: ''
    - order: 5
      text: >-
        Metrics collected and exposed with built-in Prometheus endpoint
        support
      risk: green
      rationale: ''
      mitigation: ''
- order: 3
  text: >-
    How easy is it to determine the application's health and readiness to
    handle traffic?
  explanation: >-
    How do we determine an application's health (liveness) and readiness
    to handle traffic?
  answers:
    - order: 0
```

```
      text: unknown
      risk: unknown
      rationale: ''
      mitigation: ''
    - order: 1
      text: No health or readiness query functionality available
      risk: red
      rationale: ''
      mitigation: ''
    - order: 3
      text: Basic application health requires semi-complex scripting
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 4
      text: Dedicated, independent liveness and readiness endpoints
      risk: green
      rationale: ''
      mitigation: ''
    - order: 2
      text: Monitored and managed by a custom watchdog process
      risk: red
      rationale: ''
      mitigation: ''
    - order: 5
      text: Health is verified by probes running synthetic transactions
      risk: green
      rationale: ''
      mitigation: ''
- order: 4
  text: What best describes the application's runtime characteristics?
  explanation: >-
    How would the profile of an application appear during runtime
    (examples: graphs showing CPU and memory usage, traffic patterns,
    latency)? What are the implications for a serverless application?
  answers:
    - order: 0
      text: unknown
      risk: unknown
      rationale: ''
      mitigation: ''
    - order: 1
      text: >-
        Deterministic and predictable real-time execution or control
        requirements
      risk: red
      rationale: ''
      mitigation: ''
    - order: 2
      text: >-
        Sensitive to latency (examples: voice applications, high frequency
        trading applications)
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 3
```

```
        text: Constant traffic with a broad range of CPU and memory usage
        risk: yellow
        rationale: ''
        mitigation: ''
      - order: 4
        text: Intermittent traffic with predictable CPU and memory usage
        risk: green
        rationale: ''
        mitigation: ''
      - order: 5
        text: Constant traffic with predictable CPU and memory usage
        risk: green
        rationale: ''
        mitigation: ''
  - order: 5
    text: How long does it take the application to be ready to handle traffic?
    explanation: How long the application takes to boot
    answers:
      - order: 0
        text: unknown
        risk: unknown
        rationale: ''
        mitigation: ''
      - order: 1
        text: More than 5 minutes
        risk: red
        rationale: ''
        mitigation: ''
      - order: 2
        text: 2-5 minutes
        risk: yellow
        rationale: ''
        mitigation: ''
      - order: 3
        text: 1-2 minutes
        risk: yellow
        rationale: ''
        mitigation: ''
      - order: 4
        text: 10-60 seconds
        risk: green
        rationale: ''
        mitigation: ''
      - order: 5
        text: Less than 10 seconds
        risk: green
        rationale: ''
        mitigation: ''
- order: 5
  name: Application cross-cutting concerns
  questions:
    - order: 1
      text: How is the application tested?
      explanation: >-
        Is the application is tested? Is it easy to test (example: automated
        testing)? Is it tested in production?
```

```
    answers:
      - order: 0
        text: unknown
        risk: unknown
        rationale: ''
        mitigation: ''
      - order: 1
        text: No testing or minimal manual testing only
        risk: red
        rationale: ''
        mitigation: ''
      - order: 2
        text: Minimal automated testing, focused on the user interface
        risk: yellow
        rationale: ''
        mitigation: ''
      - order: 3
        text: >-
          Some automated unit and regression testing, basic CI/CD pipeline
          testing; modern test practices are not followed
        risk: yellow
        rationale: ''
        mitigation: ''
      - order: 4
        text: >-
          Highly repeatable automated testing (examples: unit, integration,
          smoke tests) before deploying to production; modern test practices
          are followed
        risk: green
        rationale: ''
        mitigation: ''
      - order: 5
        text: >-
          Chaos engineering approach, constant testing in production
          (example: A/B testing + experimentation)
        risk: green
        rationale: ''
        mitigation: ''
  - order: 2
    text: How is the application configured?
    explanation: >-
      How is the application configured? Is the configuration method
      appropriate for a container? External servers are runtime
      dependencies.
    answers:
      - order: 0
        text: unknown
        risk: unknown
        rationale: ''
        mitigation: ''
      - order: 1
        text: >-
          Configuration files compiled during installation and configured
          using a user interface
        risk: red
        rationale: ''
```

```
      mitigation: ''
    - order: 2
      text: >-
        Configuration files are stored externally (example: in a database)
        and accessed using specific environment keys (examples: host name,
        IP address)
      risk: red
      rationale: ''
      mitigation: ''
    - order: 3
      text: Multiple configuration files in multiple file system locations
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 4
      text: >-
        Configuration files built into the application and enabled using
        system properties at runtime
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 5
      text: >-
        Configuration retrieved from an external server (examples: Spring
        Cloud Config Server, HashiCorp Consul)
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 6
      text: >-
        Configuration loaded from files in a single configurable location;
        environment variables used
      risk: green
      rationale: ''
      mitigation: ''
- order: 4
  text: How is the application deployed?
  explanation: >-
    How the application is deployed and whether the deployment process is
    suitable for a container platform
  answers:
    - order: 0
      text: unknown
      risk: unknown
      rationale: ''
      mitigation: ''
    - order: 3
      text: Simple automated deployment scripts
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 1
      text: Manual deployment using a user interface
      risk: red
      rationale: ''
      mitigation: ''
```

```
    - order: 2
      text: Manual deployment with some automation
      risk: red
      rationale: ''
      mitigation: ''
    - order: 4
      text: >-
        Automated deployment with manual intervention or complex promotion
        through pipeline stages
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 5
      text: >-
        Automated deployment with a full CI/CD pipeline, minimal
        intervention for promotion through pipeline stages
      risk: green
      rationale: ''
      mitigation: ''
    - order: 6
      text: Fully automated (GitOps), blue-green, or canary deployment
      risk: green
      rationale: ''
      mitigation: ''
- order: 5
  text: Where is the application deployed?
  explanation: Where does the application run?
  answers:
    - order: 0
      text: unknown
      risk: unknown
      rationale: ''
      mitigation: ''
    - order: 1
      text: Bare metal server
      risk: green
      rationale: ''
      mitigation: ''
    - order: 2
      text: 'Virtual machine (examples: red Hat Virtualization, VMware)'
      risk: green
      rationale: ''
      mitigation: ''
    - order: 3
      text: 'Private cloud (example: red Hat OpenStack Platform)'
      risk: green
      rationale: ''
      mitigation: ''
    - order: 4
      text: >-
        Public cloud provider (examples: Amazon Web Services, Microsoft
        Azure, Google Cloud Platform)
      risk: green
      rationale: ''
      mitigation: ''
    - order: 5
```

```
      text: >-
        Platform as a service (examples: Heroku, Force.com, Google App
        Engine)
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 7
      text: Other. Specify in the comments field
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 6
      text: Hybrid cloud (public and private cloud providers)
      risk: green
      rationale: ''
      mitigation: ''
- order: 6
  text: How mature is the containerization process, if any?
  explanation: If the team has used containers in the past, how was it done?
  answers:
    - order: 0
      text: unknown
      risk: unknown
      rationale: ''
      mitigation: ''
    - order: 1
      text: Application runs in a container on a laptop or desktop
      risk: red
      rationale: ''
      mitigation: ''
    - order: 3
      text: Some experience with containers but not yet fully defined
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 4
      text: >-
        Proficient with containers and container platforms (examples:
        Swarm, Kubernetes)
      risk: green
      rationale: ''
      mitigation: ''
    - order: 5
      text: Application containerization has not yet been attempted
      risk: green
      rationale: ''
      mitigation: ''
- order: 3
  text: How does the application acquire security keys or certificates?
  explanation: >-
    How does the application retrieve credentials, keys, or certificates?
    External systems are runtime dependencies.
  answers:
    - order: 0
      text: unknown
      risk: unknown
```

```
      rationale: ''
      mitigation: ''
    - order: 1
      text: Hardware security modules or encryption devices
      risk: red
      rationale: ''
      mitigation: ''
    - order: 2
      text: >-
        Keys/certificates bound to IP addresses and generated at runtime
        for each application instance
      risk: red
      rationale: ''
      mitigation: ''
    - order: 3
      text: Keys/certificates compiled into the application
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 4
      text: Loaded from a shared disk
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 5
      text: >-
        Retrieved from an external server (examples: HashiCorp Vault,
        CyberArk Conjur)
      risk: yellow
      rationale: ''
      mitigation: ''
    - order: 6
      text: Loaded from files
      risk: green
      rationale: ''
      mitigation: ''
    - order: 7
      text: Not required
      risk: green
      rationale: ''
      mitigation: ''
thresholds:
  red: 5
  yellow: 30
  unknown: 5
riskMessages:
  red: ''
  yellow: ''
  green: ''
  unknown: ''
builtin: true
```

7.2.2. 自定义评估问题

您可以使用 Migration Toolkit for Applications (MTA)使用自定义 YAML 语法来导入自定义评估问题。YAML 语法支持以下功能：

条件问题

YAML 语法支持根据应用程序或 archetype 的现有标签包括或排除问题，例如：

- 如果应用程序或架构有 **Language/Java** 标签，则 应用程序中使用的主 **JAVA** 框架是什么？ 问题包含在问卷中：

```
...
  questions:
    - order: 1
      text: What is the main JAVA framework used in your application?
      explanation: Identify the primary JAVA framework used in your application.
      includeFor:
        - category: Language
          tag: Java
...
```

- 如果应用程序或架构具有 **Deployment/Serverless** 和 **Architecture/Monolith** 标签，则您目前使用任何形式的容器编配？ 问题不包括在问卷中：

```
...
  questions:
    - order: 4
      text: Are you currently using any form of container orchestration?
      explanation: Determine if the application utilizes container orchestration tools
like Kubernetes, Docker Swarm, etc.
      excludeFor:
        - category: Deployment
          tag: Serverless
        - category: Architecture
          tag: Monolith
...
```

根据评估的应用程序或架构上的标签自动回答

根据应用程序或 archetype 的现有标签选择自动化答案。例如，如果应用程序或 archetype 具有 **Runtime/Quarkus** 标签，则会自动选择 **Quarkus** 回答，如果应用程序或架构有 **Runtime/Spring Boot** 标签，则会自动选择 **Spring Boot** 回答：

```
...
  text: What is the main technology in your application?
    explanation: Identify the main framework or technology used in your application.
      answers:
        - order: 1
          text: Quarkus
          risk: green
          autoAnswerFor:
            - category: Runtime
              tag: Quarkus
        - order: 2
          text: Spring Boot
          risk: green
          autoAnswerFor:
```

```
        - category: Runtime
          tag: Spring Boot
...
```

根据答案自动标记应用程序

在评估期间，如果选择了此回答，则根据回答自动应用到应用程序或架构。请注意，标签是传输的。因此，如果评估被丢弃，标签会被删除。每个标签都由以下元素定义：

- 类别 ：目标标签的类别(字符串)。
- 标签 ：目标标签的定义(字符串)。

例如，如果所选的回答是 **Quarkus**，则 **Runtime/Quarkus** 标签将应用到评估的应用程序或架构类型。如果所选的答案是 **Spring Boot**，则 **Runtime/Spring Boot** 标签将应用到评估的应用程序或架构类型：

```
...
questions:
 - order: 1
   text: What is the main technology in your application?
   explanation: Identify the main framework or technology used in your application.
   answers:
    - order: 1
      text: Quarkus
      risk: green
      applyTags:
       - category: Runtime
         tag: Quarkus
    - order: 2
      text: Spring Boot
      risk: green
      applyTags:
       - category: Runtime
         tag: Spring Boot
...
```

7.2.2.1. 自定义问问题的 YAML 模板

您可以使用以下 YAML 模板来构建自定义问题。您可以通过单击 评估问卷 页面上的 **Download YAML 模板** 来下载此模板。

例 7.2. 自定义问问题的 YAML 模板

```
name: Uploadable Cloud Readiness Questionnaire Template
description: This questionnaire is an example template for assessing cloud readiness. It serves as a guide for users to create and customize their own questionnaire templates.
required: true
sections:
 - order: 1
   name: Application Technologies
   questions:
    - order: 1
      text: What is the main technology in your application?
      explanation: Identify the main framework or technology used in your application.
```

```
    includeFor:
      - category: Language
        tag: Java
    answers:
      - order: 1
        text: Quarkus
        risk: green
        rationale: Quarkus is a modern, container-friendly framework.
        mitigation: No mitigation needed.
        applyTags:
          - category: Runtime
            tag: Quarkus
        autoAnswerFor:
          - category: Runtime
            tag: Quarkus
      - order: 2
        text: Spring Boot
        risk: green
        rationale: Spring Boot is versatile and widely used.
        mitigation: Ensure container compatibility.
        applyTags:
          - category: Runtime
            tag: Spring Boot
        autoAnswerFor:
          - category: Runtime
            tag: Spring Boot
      - order: 3
        text: Legacy Monolithic Application
        risk: red
        rationale: Legacy monoliths are challenging for cloud adaptation.
        mitigation: Consider refactoring into microservices.
  - order: 2
    text: Does your application use a microservices architecture?
    explanation: Assess if the application is built using a microservices architecture.
    answers:
      - order: 1
        text: Yes
        risk: green
        rationale: Microservices are well-suited for cloud environments.
        mitigation: Continue monitoring service dependencies.
      - order: 2
        text: No
        risk: yellow
        rationale: Non-microservices architectures may face scalability issues.
        mitigation: Assess the feasibility of transitioning to microservices.
      - order: 3
        text: Unknown
        risk: unknown
        rationale: Lack of clarity on architecture can lead to unplanned issues.
        mitigation: Conduct an architectural review.

  - order: 3
    text: Is your application's data storage cloud-optimized?
    explanation: Evaluate if the data storage solution is optimized for cloud usage.
    includeFor:
      - category: Language
```

```
        tag: Java
      answers:
       - order: 1
         text: Cloud-Native Storage Solution
         risk: green
         rationale: Cloud-native solutions offer scalability and resilience.
         mitigation: Ensure regular backups and disaster recovery plans.
       - order: 2
         text: Traditional On-Premises Storage
         risk: red
         rationale: Traditional storage might not scale well in the cloud.
         mitigation: Explore cloud-based storage solutions.
       - order: 3
         text: Hybrid Storage Approach
         risk: yellow
         rationale: Hybrid solutions may have integration complexities.
         mitigation: Evaluate and optimize cloud integration points.
  thresholds:
    red: 1
    yellow: 30
    unknown: 15
  riskMessages:
    red: Requires deep changes in architecture or lifecycle
    yellow: Cloud friendly but needs minor changes
    green: Cloud Native
    unknown: More information needed
```

其他资源

- 自定义问题字段

7.2.2.2. 自定义问题字段

每个标记为 必需的 自定义问题字段都是强制的，必须完成。否则，YAML 语法不会在上传时进行验证。字段的每个子部分在 YAML 中定义新结构或对象，例如：

```
...
name: Testing
thresholds:
    red: 30
    yellow: 45
    unknown: 5
...
```

表 7.1. 自定义问题字段

questionnaire 字段	描述
name （必需，字符串）	问卷的名称。此字段对于整个 MTA 实例必须是唯一的。
description （可选，字符串）	有关问卷的简短描述。

questionnaire 字段	描述
thresholds （必需）	应用程序或架构类型的每个风险类别的阈值定义受到该风险级别的影响。阈值可以是以下内容： ● 红帽 （必需，未签名整数）：数字百分比（例如 **30%** 的 **30** 个），问题问题可能具有的红色答案，直到风险级别被视为红色。 ● 黄色 （必需，未签名整数）：数字百分比（例如 **30%** ）的黄色答案，问问问卷在风险级别被视为黄色前可以有的回答。 ● **Unknown** （必需，未签名整数）：值值百分比（例如 **30%** 的 **30** 个），问题问题可能具有的未知答案，直到风险级别被视为未知。 更高的风险级别始终优先。例如，如果 黄色 阈值设为 **30%** 且 红色 设为 **5%**，并且应用程序或 **archetype** 的回答被设置为有 **35%** 黄色，则应用程序或架构的风险级别为红色。
riskMessages （必需）	报告中要显示的消息会为每个风险类别显示。*risk_messages* 映射由以下字段定义： ● 红帽 （必需，字符串）：报告中要针对红色风险级别显示的消息。 ● **yellow** （必需，字符串）：报告中要显示的消息，用于黄色风险级别。 ● 绿色 （必需，字符串）：报告中要显示的消息，用于绿色风险级别。 ● **Unknown** （必需，字符串）：报告中要显示为未知风险级别的消息。
部分 （必需）	问卷必须包括的部分列表。 ● **name** （必需，字符串）：要为该部分显示的名称。 ● **order** （必需，整数）：部分中问题的顺序。 ● 注释 （可选，字符串）：描述部分。 ● 问题 （必需）：属于该部分的问题列表。

questionnaire 字段	描述
	○ **order**（必需，整数）：部分中问题的顺序。 ○ 文本（必需，字符串）：要询问的问题。 ○ 解释（可选，字符串）：问题的额外说明。 ○ **includeFor**（可选）：如果目标应用程序或架构中存在此列表中的任何标签，则必须显示定义问题的列表。 ■ **category**（必需，字符串）：目标标签的类别。 ■ **tag**（必需，字符串）：目标标签。 ○ **excludeFor**（可选）：如果目标应用程序或 archetype 中存在任何标签，则必须跳过定义问题的列表。 ■ **category**（必需，字符串）：目标标签的类别。 ■ **tag**（必需，字符串）：目标标签。 ○ 答案（必需）：给定问题的回答列表。 ■ **order**（必需，整数）：部分中问题的顺序。 ■ 文本（必需，字符串）：问题的回答。 ■ **risk**（必需）：当前回答的风险级别(red, yellow, green, 或 unknown)。 ■ 比例（可选，字符串）：对于被视为风险的答案而言，合理性。 ■ 缓解方案（可选，字符串）：对于答案所指示的风险，潜在的缓解策略的说明。 ■ **applyTags**（可选）：如果选择了此回答，则自动应用到评估的应用程序或架构标签列表。 ● **category**（必需，字符串）：目标标签的类别。 ● **tag**（必需，字符串）：目标标签。 ■ **autoAnswerFor**（可选，list）：在评估应用程序或架构类型时，将自动选择此回答的标签列表。 ● **category**（必需，字符串）：目标标签的类别。 ● **tag**（必需，字符串）：目标标签。

questionnaire 字段	描述

其他资源

- 自定义问问题的 YAML 模板

7.3. 管理评估问题

通过使用 MTA 用户界面，您可以对评估问题执行以下操作：

- 显示问卷.您还可以选择回答选项以及相关的风险权重。
- 将问卷导出到您系统上的所需位置。
- 从您的系统导入问卷。

警告

导入的问题的名称必须是唯一的。如果 YAML 语法中定义的名称(name: <name of questionnaire>)被重复，则导入将失败，并显示以下错误消息：**UNIQUE constraint failed: issuesnaire.Name**。

- 删除评估问卷。

警告

当您删除问卷时，所有架构类型中使用的应用程序都会删除其答案。

重要

您不能删除 Legacy Pathfinder 默认问题。

流程

- 根据您的场景，执行以下操作之一：
  - 显示评估问卷：
    1. 在 Administration 视图中，选择 assessment questionnaires。
    2. 点击 Options 菜单(⋮)。
    3. 为您要显示的问题选择 View。

4. 可选：点击问题左侧的箭头，以显示回答选项及其风险权重。

- 导出评估问卷：
  1. 在 **Administration** 视图中，选择 **assessment questionnaires**。
  2. 选择所需的问卷。
  3. 点击 **Options** 菜单( )。
  4. 选择 **Export**。
  5. 选择下载位置。
  6. 点击 **Save**。
- 导入评估问题：
  1. 在 **Administration** 视图中，选择 **assessment questionnaires**。
  2. 点 **Import questionnaire**。
  3. 点 **Upload**。
  4. 导航到问卷的位置。
  5. 点 **Open**。
  6. 点 **Import** 导入所需的问题。
- 删除评估问题：
  1. 在 **Administration** 视图中，选择 **assessment questionnaires**。
  2. 选择您要删除的问卷。
  3. 点击 **Options** 菜单( )。
  4. 选择 **Delete**。
  5. 在问卷 的名称 中输入 来确认删除。

其他资源

- 默认评估问题
- 自定义评估问题

7.4. 评估应用程序

您可以通过执行应用程序评估来估算准备应用程序进行容器化的风险和成本。您可以使用 **assessment** 模块评估应用程序，并显示当前保存的评估。

Migration Toolkit for Applications (MTA)根据与应用程序相关的一组问题（如依赖项）评估应用程序。

要评估应用程序，您可以使用默认的 **Legacy Pathfinder MTA** 问卷调查或导入您的自定义 问题。

重要

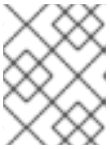
您一次只能评估一个应用程序。

先决条件

- 您已登录到 MTA 服务器。

流程

1. 在 MTA 用户界面中，选择 Migration 视图。
2. 单击左侧菜单栏中的 Application inventory。可用的应用程序列表会出现在主窗格中。
3. 选择您要评估的应用程序。
4. 点击行右侧的 Options 菜单( )，然后从下拉菜单中选择 Assess。
5. 从可用问卷列表中，单击 所需问卷。
6. 从列表中选择 Stakeholders 和 Stakeholder groups 来跟踪为未来参考提供评估的人员。



注意

您还可以在 Migration 视图的 Controls 窗格中添加 Stakeholder Groups 或 Stakeholders。如需更多信息，请参阅查看实例。

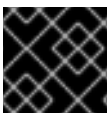
7. 点击 Next。
8. 回答 每个应用程序评估 问题，然后单击下一步。
9. 点 Save 查看评估，然后执行 检查应用程序 中的步骤。

其他资源

- 默认评估问题
- 自定义评估问题
- 管理评估问题

7.5. 检查应用程序

您可以使用 Migration Toolkit for Applications (MTA)用户界面来确定每个应用程序的迁移策略和工作优先级。



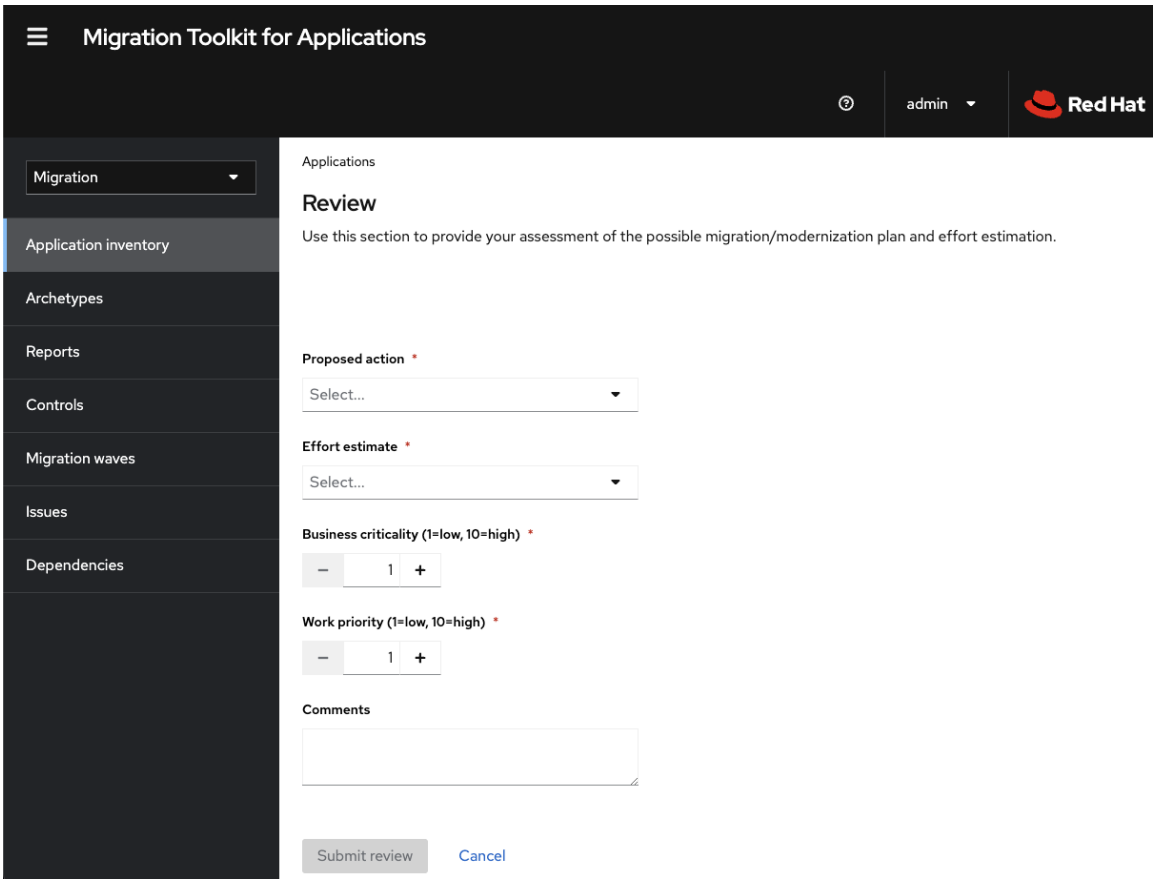
重要

一次只能查看一个应用程序。

流程

1. 在 Migration 视图中，点 Application inventory。

2. 选择您要查看的应用程序。
3. 通过执行以下操作之一查看应用程序：
   - 在评估应用程序时，点 **Save and Review**。如需更多信息，请参阅 评估应用程序。
   - 点击行右末尾的 **Options** 菜单( )，然后从下拉菜单中选择 **Review**。应用程序 **Review** 参数会出现在主窗格中。



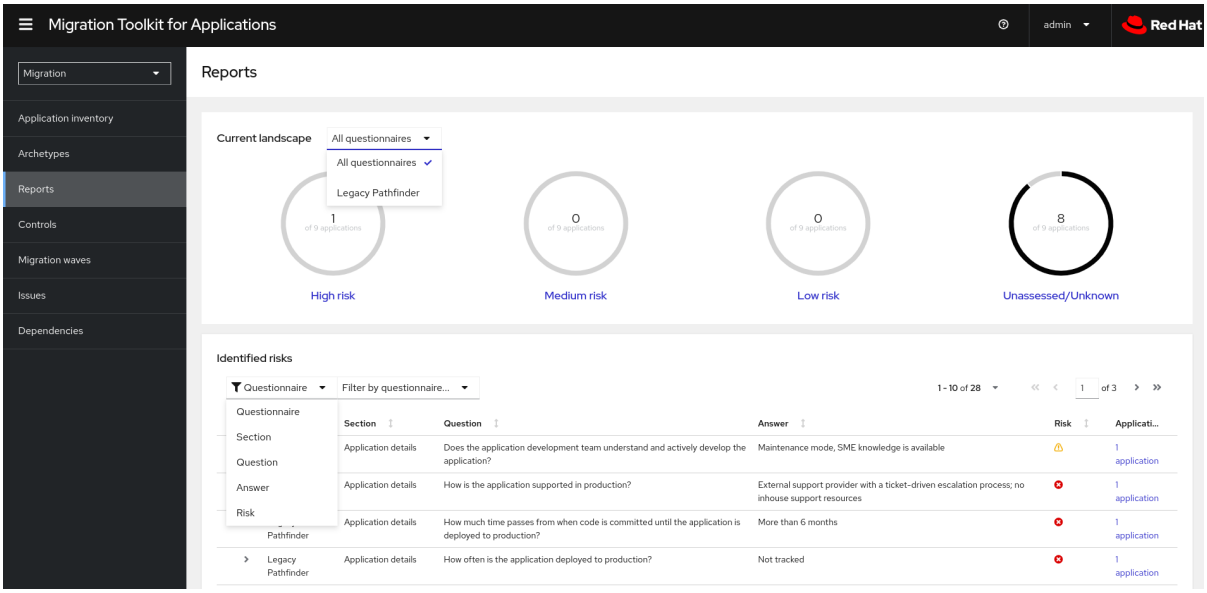
4. 点 **Proposed action** 并选择操作。
5. 点 **Effort estimate** 并设置对所选问卷执行评估所需的工作量程度。
6. 在 **Business criticality** 字段中输入应用程序对业务的关键。
7. 在 **Work priority** 字段中，输入应用程序的优先级。
8. 可选：在注释字段中输入评估问卷 注释。
9. 点 **Submit review**。
   Review 中的字段现在在 **Application details** 页面中填充。

7.6. 查看评估报告

MTA 评估报告显示从多个问题为多个应用程序获得的数据的聚合评估。

流程

1. 在 **Migration** 视图中，点 **Reports**。此时会显示所有应用程序的聚合评估报告。



2. 根据您的场景，执行以下操作之一：

- 显示来自特定问卷的数据的报告：

  a. 从报告的 **Current landscape** 窗格中的所有问卷下拉列表中，选择所需的问卷。默认情况下，会选择所有问题。

  b. 在 报告的识别风险 窗格中，根据应用程序名称、风险级别、问题部分、问题和答案对显示的列表进行排序。

- 显示特定应用程序的报告：

  a. 点报告的 **Identified risk** 窗格中的**Applications** 列中的链接。此时会打开 **Application inventory** 页面。链接中包含的应用程序以列表的形式显示。

  b. 点所需应用程序。此时会打开 **assessment** 侧窗格。

  - 要查看应用程序评估的风险级别，请打开 **Details** 选项卡。
  - 要查看评估详情，请打开 **Reviews** 选项卡。

7.7. 标记应用程序

您可以将各种标签附加到您要分析的应用程序。您可以使用标签对应用程序进行分类，并即时识别应用程序信息，例如应用程序中使用的应用程序类型、数据中心位置和技术。您还可以使用标记将 **archetypes** 与应用程序关联，以进行自动评估。有关 **archetypes** 的更多信息，请参阅使用 **archetypes**。

在分析过程中， 可以随时 手动执行 标记。

注意

并非所有标签都可以自动分配。例如，分析只能基于其技术标记应用程序。如果要使用部署的数据中心的位置标记应用程序，则需要手动标记应用程序。

7.7.1. 创建应用程序标签

您可以为 **MTA** 评估或分析的应用程序创建自定义标签。

流程

1. 在 Migration 视图中，点 Controls。
2. 单击 标签选项卡。
3. 点 Create tag。
4. 在打开的对话框中的 Name 字段中输入标签的唯一名称。
5. 单击 Tag category 字段，再选择与标签关联的类别标签。
6. 点 Create。
7. 可选：编辑创建的标签或标签类别：
   - 编辑标签：
     i. 在 Tags 选项卡下的标签类别列表中，打开所需类别中的标签列表。
     ii. 从下拉菜单中选择 Edit，然后在 Name 字段中选择标签名称。
     iii. 单击 Tag category 字段，再选择与标签关联的类别标签。
     iv. 点击 Save。
   - 编辑标签类别：
     i. 在 Tags 选项卡下，选择定义的标签类别，再单击 Edit。
     ii. 在 Name 字段中编辑标签类别的名称。
     iii. 编辑类别的 Rank 值。
     iv. 点 Color 字段，为标签类别选择一个颜色。
     v. 点击 Save。

7.7.2. 手动标记应用程序

您可以在应用程序分析前或之后手动标记应用程序。

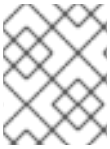
流程

1. 在 Migration 视图中，点 Application inventory。
2. 在所需应用程序所在的行中，点 Edit ( )。此时会打开 Update application 窗口。
3. 从 Select a tag(s) 下拉列表中选择所需的标签。
4. 点击 Save。

7.7.3. 自动标记

MTA 可以根据应用程序分析自动向应用程序添加标签。自动标记在处理大型应用程序组合时特别有用。

默认情况下启用自动标记应用程序。您可以通过在 Analysis 配置 向导的 Advanced 部分取消选择 Enable automated tagging 复选框来禁用自动标记。



注意

要自动标记应用程序，请确保在运行应用程序分析 *前* 选择了 Enable automated tagging 复选框。

7.7.4. 显示应用程序标签

您可以显示附加到特定应用程序的标签。

注意

您只能在运行应用程序分析后显示自动附加的标签。

流程

1. *在 Migration 视图中，点 Application inventory。*
2. *点所需应用程序的名称。此时会打开侧窗格。*
3. *单击 标签选项卡。此时会显示附加到应用程序的标签。*

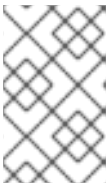
7.8. 使用 ARCHETYPES

archetype 是一组具有常见特征的应用程序。您可以使用 archetypes 同时评估多个应用程序。通过使用 archetypes，Migration Toolkit for Applications (MTA) 可以应用问题，其中包含适用于常见应用程序特性的问题。

应用程序架构类型由标准标签和应用程序税务定义。每个 archetype 定义 评估模块 如何根据该架构类型中定义的特征评估应用程序。如果应用的标签与 archetype 的条件标签匹配，则应用程序与 archetype 关联。

创建架构类型由一系列 标签、利益相关者和利益相关者 组定义。标签包括以下类型：

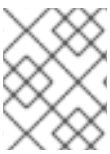
- *条件 标签是 archetype 需要将应用作为成员包含的标签。*



注意

如果 archetype 条件标签仅与应用程序部分匹配，则此应用不能是 archetype 的成员。例如，如果应用程序只有一个标记，但标准标记包含标记 一个 AND b ， 则应用程序 a 不是架构类型的成员。

- *archetype 标签是应用到 archetype 实体的标签。*



注意

与 archetype 关联的所有应用程序都会继承 评估，并从这些应用程序所属的 archetype 组进行检查。这是默认设置。您可以通过完成单个评估和审核来覆盖应用程序的继承。

7.8.1. 创建 archetype

当您创建 archetype 时，如果此应用具有与 archetype 标签匹配的标签，则清单中的应用程序会自动与该架构类型关联。

流程

1. 打开 MTA web 控制台。
2. 在左侧菜单中，单击 Archetypes。
3. 单击 Create new archetype。
4. 在打开的形式中，为新 archetype 输入以下信息：
   - 名称 ：新 archetype 的名称（必需）。
   - 描述 ：新 archetype 的描述（可选）。
   - Standard Tags: 标记将评估的应用程序与 archetype （必需）关联。如果更新了条件标签，则会再次触发与 archetype 关联的应用程序的流程。
   - arch etype Tags: 标签，标记应用程序中的 archetype 评估（必需）。
   - 利益相关者：涉及应用程序开发和迁移的特定拥有者（可选）。
   - 利益相关者组 ：涉及应用程序开发和迁移的利益相关者组（可选）。
5. 点 Create。

7.8.2. archetype 评估

当所有需要的问题被回答时，会考虑一个架构类型。

当被检查一次，即使多个问题已被标记为必需，也会被视为架构类型。

如果应用程序与 archetypes 关联，则当所有关联的架构类型被评估时，这个应用程序将被视为评估。

7.8.3. 删除 archetype

删除架构类型会删除任何关联的评估。所有关联的应用程序都移至 Unassessed 状态。

7.9. 分析应用程序

您可以使用 Migration Toolkit for Applications (MTA) 用户界面来配置和运行应用程序分析。分析决定应用程序中的哪些特定行必须修改，然后才能迁移或现代化应用程序。

7.9.1. 配置并运行应用程序分析

您可以一次分析多个应用程序，在同一分析中有多个转换目标。

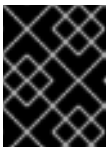
流程

1. 在 Migration 视图中，点 Application inventory。
2. 选择您要分析的应用程序。

3. 检查分配给应用程序的凭据。
4. 点 **Analyze**。
5. 从列表中选择 **Analysis 模式**：
   - 二进制
   - 源代码
   - 源代码和依赖项
   - 上传本地二进制文件。只有在分析单个应用程序时会出现这个选项。如果选择了这个选项，系统会提示您上传本地二进制。将文件拖动到提供的区域或单击 **Upload**，然后选择要上传的文件。
6. 点击 **Next**。
7. 为分析选择一个或多个目标选项：
   - 应用程序服务器迁移到以下平台之一：
     - **JBoss EAP 7**
     - **JBoss EAP 8**
   - 容器化
   - **Quarkus**
   - **OracleJDK 到 OpenJDK**
   - **OpenJDK**。使用这个选项升级到以下 **JDK 版本之一**：
     - **OpenJDK 11**
     - **OpenJDK 17**
     - **OpenJDK 21**
   - **Linux**。使用这个选项来确保没有将 **Microsoft Windows** 路径硬编码到应用程序中。
   - **Jakarta EE 9.**使用这个选项从 **Java EE 8** 迁移。
   - **Spring Boot on Red Hat Runtimes**
   - **Open Liberty**
   - **camel.**使用这个选项从 **Apache Camel 2** 迁移到 **Apache Camel 3** 或从 **Apache Camel 3** 迁移到 **Apache Camel 4**。
   - **Azure** 应用程序服务
8. 点击 **Next**。
9. 选择以下范围选项之一以更好地关注分析：
   - 仅限应用和内部依赖项。

- 应用程序及所有依赖项，包括已知的开源库。
- 选择要手动分析的软件包列表。如果选择了这个选项，请键入文件名并点添加。
- 排除软件包。如果选择了这个选项，请键入软件包的名称并点 **Add**。

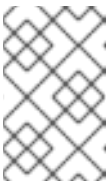
10. 点击 **Next**。
11. 在 **Advanced** 中，您可以通过选择 **Manual** 或 **Repository** 模式，将额外的自定义规则附加到分析中：
    - 在 **Manual** 模式中，单击 **Add Rules**。拖动相关文件或从其目录中选择文件，然后单击 **Add**。
    - 在 **Repository** 模式中，您可以从 **Git** 或 **Subversion** 存储库添加规则文件。



重要

如果您已将迁移目标附加到分析中，附加自定义规则是可选的。如果您还没有附加任何迁移目标，则必须附加规则。

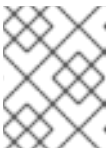
12. 可选：设置以下选项之一：
    - 目标
    - 源
    - 排除的规则标签。没有处理具有这些标签的规则。根据需要添加或删除。
    - 启用自动标记。选中复选框，以自动将标签附加到应用。默认选择此复选框。



注意

只有运行分析后才会显示自动附加的标签。

您可以手动将标签附加到应用程序，而不是启用自动标记或额外启用。



注意

分析引擎将标准规则用于一组全面的迁移目标，但如果目标未包含或是自定义框架，则可以添加自定义规则。只有手动上传的自定义规则文件才会被验证。

13. 点击 **Next**。
14. 在 **Review** 中，验证分析参数。
15. 点 **Run**。
    当 **MTA** 下载要执行的容器的镜像时，分析状态会被调度。下载镜像后，状态会更改为 ***In-progress***。

注意

分析需要几分钟的时间根据应用程序的大小以及集群容量和资源来运行。

提示

***MTA** 依赖于 **Kubernetes** 调度功能来确定基于集群容量创建多少分析器实例。如果选择多个应用程序进行分析，则一次只能置备一个分析器。使用更多集群容量时，可以并行执行更多分析过程。*

16. *分析完成后，点 **Report** 链接来查看分析的结果。*

17. *可选：显示分析的详情：*

    a. *点击 **Options** 菜单(*  *)。*

    b. *选择 **Analysis details**。您可以选择 **YAML** 或 **JSON** 格式。*

7.9.2. 查看分析报告

***MTA** 分析报告包含多个部分，包括应用程序使用的技术列表、应用依赖项以及必须更改的代码行，才能成功迁移或现代化应用程序。*

*有关 **MTA** 分析报告内容的更多信息，请参阅 检查报告。*

流程

1. *在 **Migration** 视图中，点 **Application inventory**。*

2. *通过完成的分析来扩展应用程序。*

3. *点 **Reports**。*

4. *点 **dependencies** 或 **source** 链接。*

5. 点选项卡查看报告。

7.9.3. 下载分析报告

为方便起见，您可以下载分析报告。请注意，默认情况下这个选项被禁用。

流程

1. 在 **Administration** 视图中，点 **General**。
2. 在 运行分析后切换 **Allow report to download**。
3. 进入 **Migration** 视图，再点 **Application inventory**。
4. 打开您运行分析的应用程序页面。
5. 点 **Reports**。
6. 点 **HTML** 或 **YAML** 链接：
   - 点击 **HTML** 链接，您可以下载压缩的 **analysis-report-app-<application_name>.tar** 文件。提取此文件会创建一个名称与应用相同的文件夹。
   - 点击 **YAML** 链接，您可以下载未压缩的 **analysis-report-app-<application_name>.yaml** 文件。

7.10. 创建自定义迁移目标

具有 **admin** 权限的架构师或用户可以创建和维护与自定义迁移目标相关的自定义规则集。架构师可以上传自定义规则文件，并将它们分配到各种自定义迁移目标。然后，可以在分析配置向导中选择自定义迁移目标。

通过使用可用的自定义迁移目标，您可以避免为每个分析运行配置自定义规则。这简化了非管理员用户

或第三方开发人员的分析配置和执行。

先决条件

- *您以具有 **admin** 权限的用户身份登录。*

流程

1. *在 **Administration** 视图中，点 **Custom migration targets**。*
2. *点 **Create new**。*
3. *输入目标的名称和描述。*
4. *在 **Image** 部分中，上传目标图标的图形文件。该文件可以是 **PNG** 或 **JPEG** 格式，最多为 **1 MB**。如果您没有上传任何文件，则使用默认图标。*
5. *在 **Custom rules** 部分中，从存储库选择 **Upload manually** 或 **Retrieve** ：*
   - *如果您选择了 **Upload manually**，请上传或拖放本地驱动器中所需的规则文件。*
   - *如果您从 存储库选择了 **Retrieve**，请完成以下步骤：*
     i. *选择 **Git** 或 **Subversion**。*
     ii. *输入 **Source repository**、**Branch** 和 **Root** 路径 字段。*
     iii. *如果存储库需要凭证，请在 **associated d credentials** 字段中输入这些凭证。*
6. *点 **Create**。*

*新迁移目标会出现在 **Custom migration targets** 页面中。现在，非管理员用户可在*

***Migration** 视图中使用。*

*更新于 **2024-05-23***