Modernize your Java applications faster with IBM Application Modernization Accelerator

Instructor — led lab guide

Author name
Paul Barry

paulbarr@ibm.com

Architect, Application Modernization Tooling

Table of Contents

1. Introduction 5

1.1. About this hands-on lab 6

2. Using the lab environment 6

2.1. Logging in 6

2.2. Accessing Software 7

2.3. Sending Text 7

2.4. Getting this guide 7

3. Modernizing your whole java estate with Application Modernization Accelerator (AMA) 7

3.1. Launching Application Modernization Accelerator 8

3.2. Visualization and Assessment 8

3.2.1. Taking the tour 8

3.2.2. Analyzing ACME 8

3.2.3. Analyzing ModResorts (and common code) 9

3.3. Download the migration plan. 10

4. Modernizing the code with AMA Dev Tools 11

4.1. Modernizing the ModResorts source code 11

4.2. Testing modernized source code 11

4.3. Modernizing common code 12

4.4. Using the modernized common code 13

4.5. Modernization Guidance 14

5. Extra Credit (Optional) 14

5.5. Creating an IBM Cloud account 14

5.2. Creating an IBM Cloud API key 15

5.3. Creating a configuration in IBM Cloud 15

6. Getting help and troubleshooting 16

6.3. Reach out to the Lab instructor 16

6.4. Common troubleshooting tips 16

3. Lab Reference Materials 17

Notices and disclaimers

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

This document is current as of the initial date of publication and may be changed by IBM at any time. Not all offerings are available in every country in which IBM operates.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information.

This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity. IBM products and services are warranted per the terms and conditions of the agreements under which they are provided. The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

IBM products are manufactured from new parts or new and used parts.
In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled,
isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to ensure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Notices and disclaimers (Continued)

Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

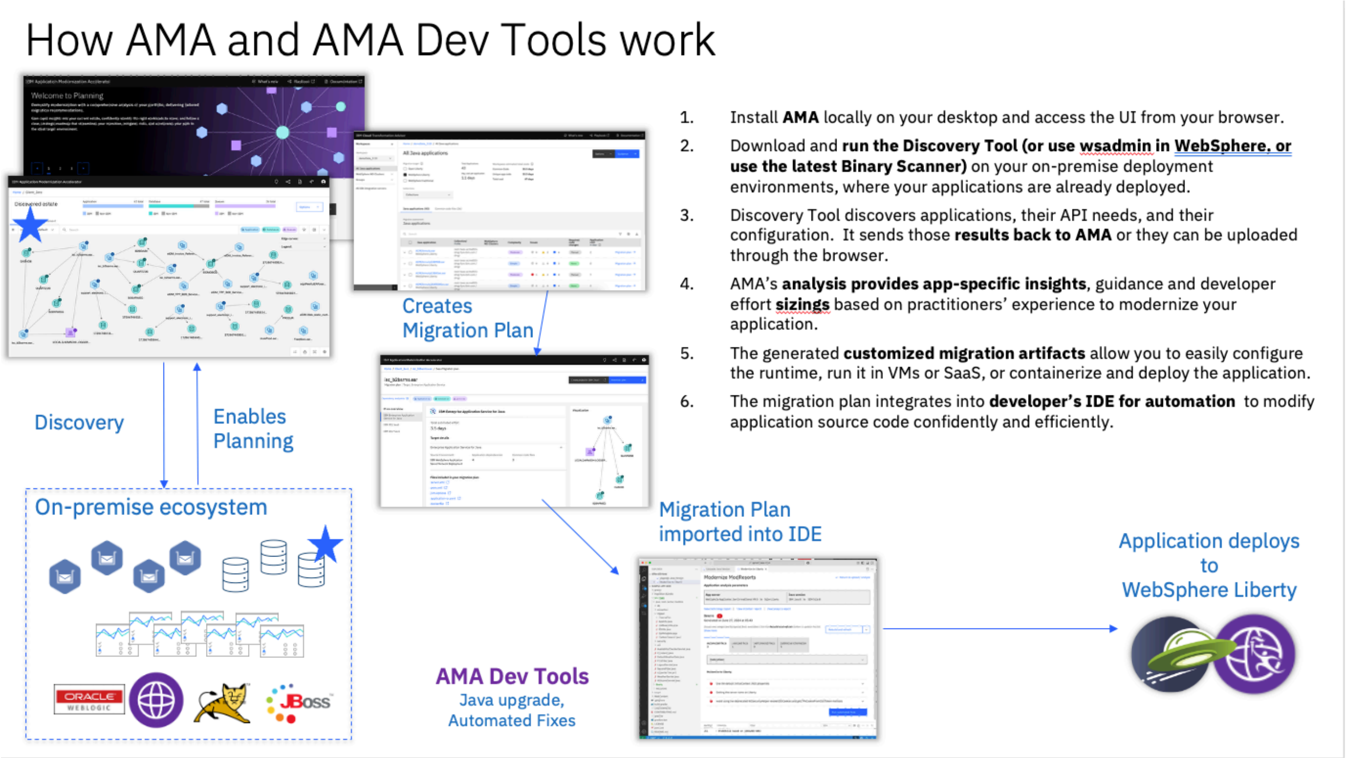
1. Introduction

Application Modernization Accelerator is a modernization tool designed to scan and discover your entire Java application estate. The applications are scanned in place to allow for the collection of their configuration so that a detailed picture of the connections between applications, database and queues can be discovered.

Analysis is provided to identify the challenges in going to a list of different runtime destinations and common code across the estate is identified to ensure the work of modernization is only done once, and not multiple times. Guidance is provided on how to build modernization momentum by concentrating on the applications that share common code, so code changes help to modernize as many applications as possible as quickly as possible.

The analysis and configuration collected during the scan is then used to generated migration artifacts that are customized for each application to accelerate the modernization of the application to the desired runtime destination.

The estate wide view of all stages of the process, from scanning, to analysis to modernization implementation allows for accurate planning, risk assessment and evaluation of the return-on-investment for each potential destination.



How AMA and AMA Dev Tools work

1. Install **AMA** locally on your desktop and access the UI from your browser.
2. Download and **run the Discovery Tool (or use wsadmin in WebSphere, or use the latest Binary Scanner)** on your on-premise deployment environments, where your applications are already deployed.
3. Discovery Tool discovers applications, their API needs, and their configuration. It sends those **results back to AMA** or they can be uploaded through the browser.
4. AMA's **analysis provides app-specific insights**, guidance and developer effort **sizings** based on practitioners' experience to modernize your application.
5. The generated **customized migration artifacts** allow you to easily configure the runtime, run it in VMs or SaaS, or containerize and deploy the application.
6. The migration plan integrates into **developer's IDE for automation** to modify application source code confidently and efficiently.

In this lab the install of AMA and the execution of the Discovery Tool has already been completed.

Application Modernization Accelerator developer tools are designed to run on your application source code and modernize both the Java runtime and the Java version

Java runtime modernization allows you to analyze your legacy enterprise Java application and modernize it to use a more lightweight, flexible, and efficient runtime. Receive a prescriptive plan that describes the changes that are needed to modernize your application with a detailed assessment of complexity and required development effort. Use automation to quickly implement code and configuration changes.

Java version modernization allows you to identify changes required to upgrade Java code and automatically update the relevant application code. Receive a prescriptive plan that describes the changes that are needed to upgrade your application with a detailed assessment of complexity and required development effort. Use automation to quickly implement code changes.

1.1. About this hands-on lab

The main objective of this hands-on lab is to present an end-to-end overview of the capabilities of Application Modernization Accelerator and to explicitly show how the AMA developer tools can quickly and easily modernize Java applications.

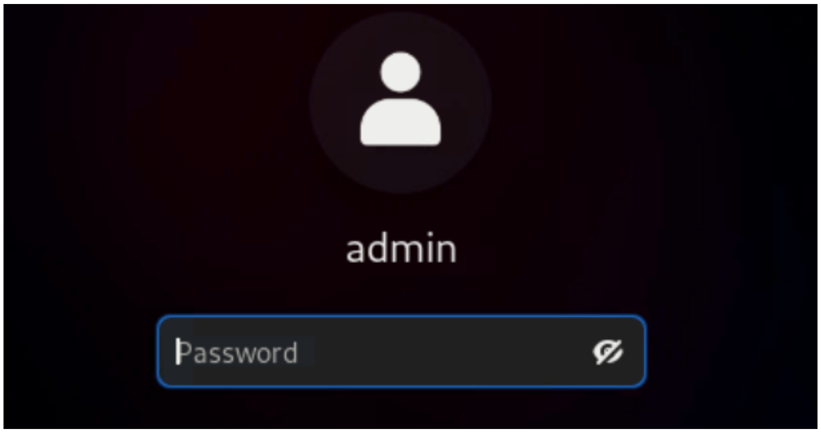
Modernization is a complex area with many conflicting requirements. In essence the goal is always to modernize as much as possible as quickly as possible, but in practice modernization is “One size fits none.” There is rarely any return on investment to modernize all applications to the very latest technologies and architectures. AMA is designed to allow you to develop a strategy that best suits your business needs and then to accelerate its implementation.

2. Using the lab environment

The lab environment has already been created with all the necessary data, tools and software that you require.

2.1. Logging in

Click anywhere on the screen to get the login option.



Use the following credentials:

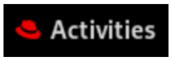
- User: admin

- Password: IBMDem0s

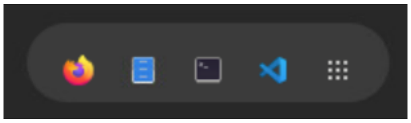
2.2. Accessing Software

Use the activities to access the available software

1. Click **Activities** at the top left



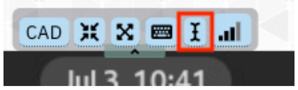
2. The software you will use appears in the bar at the bottom of the screen



2.3. Sending Text

When working with the VM the standard copy and paste option only work within the VM. Use the Send Text function if you are copying text from an external source into the VM

1. Click in the place where you want to send the text to
2. Click the Send Text icon at the top of the screen



3. Press **Ctrl+V** to copy in the text
4. Click **Fast Send Text**

2.4. Getting this guide

This guide is available at https://github.com/paulbarr/tx25-lab2639

3. Modernizing your whole java estate with Application Modernization Accelerator (AMA)

AMA is designed to accelerate the modernization of your whole Java estate. In this lab you will take on the role of the modernization architect who has been tasked with modernizing InstantPayments, an aging application infrastructure. While the whole estate needs to be modernized there are two critical business applications that have been identified as needing to be modernized first. These business applications are:

- ACME
- ModResorts

3.1. Launching Application Modernization Accelerator

The first step is to start the AMA instance via the script provided

1. Click **Activities** at the top left, and open **Terminal**
2. Run the following command: **./startAMA.sh**

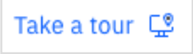
AMA will start. It takes about 1 minute to become available. When it has fully started a URL is provided. For simplicity in this lab AMA is already available as a bookmark in Firefox.

3.2. Visualization and Assessment

The purpose of the visualization is to see each of your applications and their connection. It is very useful in understanding how your applications are related to each other via these connections.

3.2.1. Taking the tour

To see the capabilities of the visualization we have a guided tour available.

1. Click **Activities** at the top left, and open **Firefox**. **AMA** should be the default page that opens. If not click on **AMA** in the bookmark bar.
2. Click on the **TX2025** workspace.
3. Click **Take a tour**  in the top right-hand part of the screen.

3.2.2. Analyzing ACME

Having completed the tour you we will now focus on the business applications that are critical to modernize. The first business application is **ACME**.

1. Click on **Visualization** tab.
2. In the Graph Search bar above the visualization, type **ACME**. All the **ACME** applications and the nodes connected to them become highlighted. The ACME business application is made up of 6 applications, supported by three messaging queues and one database.
3. Click on the **ANNUITY** database node.
4. The panel on the right-hand side provides summary information about the selected node - in this case, its **ANNUITY** database. We can see that it is a **DB2** database type. Click on **Details** to get more information of the node.

5. You are brought to the database details page. In the **Overview** section, You can see that the **Total application cost** to modernize this database for deployment to the cloud is estimated at **8.5** days of effort. Click the **back button** in your browser.

6. The ACME business capability is made up of a series of connected apps, messaging queues and a database. Connected groups of applications like this can also be explored from the dynamically generated **Connected groups**. Click in the **expander arrow** › in the top left of the screen. Click on **Connected groups** to expand the list of groups.

7. Click on **Connected_Group_A(6).** This highlights the ACME group, which was automatically detected as a group of interrelated applications. The number in brackets represented the number of applications in the group.

8. Click on **Connected_Group_B(7).** You will see another group of applications highlighted. Zoom in on that group by clicking **+** in the mini map and you will see the applications have different names and are not obviously related as the ACME applications are.

   This group is representative of what happens when infrastructure grows organically. Some applications in this group are likely to be genuinely related (such as roller-ui.ear and roller.ear) and some are likely to be related only because they use common infrastructure (such as multiple applications using txdb). Irrespective of why they are related it is critical during modernization to know how your whole estate relates so that modernization can be planned in such a way as to minimise disruption to the existing running estate.

9. Click back to **Connected_Group_A(6)** and click on the **Assessment** tab in the top left. This will show you the modernization effort required for each of the applications in the ACME group. The overall effort is estimated at 7.5 days, and this will be in addition to the modernization effort required for the database. We will explore this view in more detail later in the lab.

   We have learned a lot of useful information about what will be involved in modernizing the ACME business application. It has many connections, moving parts and will require the updating of code in multiple code repositories. AMA will accelerate this process, but it is not an ideal business application to start with because of its size and breath.

3.2.3. Analyzing ModResorts (and common code)

We will now focus on the other critical business application called **ModResorts**.

1. Click on **Applications**

in the left nav to return to the view that has the whole estate.

2. Click on the **Visualization** tab.

3. In the **Overview** panel on the right-hand side click on **Apps** on the switcher, and type **mod** at the search bar.

4. Now click on the name in the list: **modresorts-2_3_0_war.ear**.

5. The application node is highlighted in the visualization.

   This application has no connections to databases or messaging queues which simplifies the modernization and deployment of this application.

6. In the panel on the right you can see that the application is listed as having **Moderate** complexity, and scroll down to the **Summary** section, it shows an estimated effort of **0.5 days**. Click on the **Details** button.

7. The application is marked with a Complexity of **Moderate** and with Code changes **Automated**.

   This means that the application required code changes before it will run on EASeJ. However, these codes changes are fully automated and can be executed at the click of a button.

8. On the left-hand side click on **Required code changes**. The screen will scroll down and you will see the configuration necessary to automate the code changes. The good news is that we will use **AMA Dev Tools** to do this automatically, so you will not need to manually apply this configuration!

9. Scroll down to the **Issues** section. The issues are divided into **Common Code Issues** and **Unique Code Issues**. Common Code is code that appears in applications other than this one, so modernizing that code will help other applications as well.

10. Expand the **Technology Issues** for **Common Code**, and expand the issue to read the the issue detail. In the **Issue help**, you will see that the line 'These methods are deprecated in traditional WebSphere……They are not available on Liberty', so it will have to be replaced with something else.

11. Open the **Technology Issues** for **Unique Code** and review the two issues.

    *Issues marked as **Critical** ⛔ indicates work that needs to be done before the application will run on EASeJ. Issues marked as **Informational** ℹ indicates code that we expect to work, but behave in unexpected ways. We recommend that you address these issues only if problems are found during testing.*

12. Scroll down to the **Common code files**. You can see that there is a common library called **env-config-2.2.jar** that is used by two applications. Click on **env-config-2.2.jar**.

13. The **env-config-2.2.jar** details opens. Scroll to **Common code file** section at the bottom of the screen and you can see **ReadinessAgent-1_2_war.ear** and **modresorts-2_3_0_war.ear** both use the exact same copy of this file.

    The identification of common code is a critical part of accelerating the modernization journey. Most organization make extensive reuse of code. Identifying common code avoids modernization work being done multiple times and helps significantly with planning by providing a much more accurate respresentation of the work that is to be done.

14. Click the back button in the browser to return to the **modresorts-2_3_0_war.ear** details page.

    ModResorts has no connections and all the code changes required are fully automated. This makes it an excellent first application to modernize.

3.3. Download the migration plan.

We will now modernize the ModResorts application.

1. Click on the **View migration plan** button in the top right-hand corner.

2. For now we will skip to making the necessary code changes and come back and look at this page in detail later in the lab. Click the **Download plan** button. Save the file that is generated.

4. Modernizing the code with AMA Dev Tools

We will now modernize the ModResorts application using the AMA Dev Tools.

4.1. Modernizing the ModResorts source code

Using the migration bundle provided we will update the source code using Visual Studio Code

1. Now we will work with the source code of the ModResorts application in Visual Studio Code. Click on **Activities** Activities and choose to open **Visual Studio Code**
2. The **MOD-RESORTS-TX25** project should open automatically. If not choose **File > Open Recent** and select it.
3. VSCode already has the AMA Dev Tools installed and we will use those to modernize to Liberty. Right click on any file in the project and choose: **Modernize Java Applications** > **Modernize to Liberty**.
4. A panel appears that provides information about modernization. Scroll down and choose **Upload migration plan** Upload migration plan. Select the ModResorts migration plan that you downloaded above.
5. The tool will select the **server.xml** file to import into your project by default. Press **Proceed**.
6. The migration bundle is imported and the source code analysed. There are 3 issues found and all have automated fixes. Expand the second issue and note that this issue has been detected in external code.
7. Scroll to the bottom and click **Run automated fixes.** This will resolve the issues detected in this source code, but it cannot resolve issue in external source code.

   Note: There may be Information issues remaining. These issues may impact your testing are for reference only. You can deploy to the new runtime with these issues

8. If the left navigation click on the **Source Control** icon. 4 files have been modified by the import of the migration bundle and execution of the automated fixes. You can review the changes that have been made.

   Note: The code used in this lab has been simplified to highlight the changes that have been made.

9. We will run the analysis again to confirm that the issues have been resolved. Scroll up and click **Rebuild and refresh** Rebuild and refresh. There is still 1 issue remaining because it is in external code.

4.2. Testing modernized source code

There is still one outstanding issue in the external code but we will test to see if it is necessary to modernize it. The Liberty Plugin has already been installed to allow you to run Liberty locally.

1. In VSCode, in the left navigation click on **Explorer** and then open the **Liberty Dashboard** ∨ LIBERTY DASHBOARD. There is one entry called **modresorts**.

2. Right click on it and choose **Start**. The ModResorts application will be started in a local Liberty server so you can test it.

3. The application is available at http://localhost:9080/resorts. There is also a bookmark for it in Firefox called

4. The application opens and everything looks like it is working! But is it……….

5. Click the **Where to?** dropdown and select **Cork, Ireland**. Aaarrgghhhh – there are lots of errors. It turns out that the common code is being used and does need to be modernized.

   In some cases, applications will not start without modernization, and in other cases applications will start but certain capabilities will not work. That is why it is so important to use tools to help you identify potnatil modernization issues.

6. In the left navigation, Click on **Explorer** , and open the **Liberty Dashboard** ∨ LIBERTY DASHBOARD**,** right click on **modresorts** and choose **Stop.** This will stop the local Liberty server.

4.3. Modernizing common code

We have identified external common code that ModResorts is using that needs to be modernized. This code is in **env-config-2.2.jar**. We will now update that code.

1. In VSCode click **File** > **Open Recent** and select **env-config**.

2. To modernize this to Liberty right click on any file and choose: **Modernize Java Applications** > **Modernize to Liberty**

3. We don't have a migration plan this time, so scroll to the bottom and expand **Analyze Application**. Click **Start analysis**.

4. For Application Server set the following values.

   a. **Source**: WebSphere Application Server traditional V9.0

   b. **Target**: WebSphere Liberty

5. For Java SE version set the following values.

   a. **Source**: IBM Java 8

   b. **Target**: IBM Java 8

   For this lab we will keep the Java level the same. Once modernized to a new runtime the application can then be modernized to a newer Java version

6. Now click **Build and Analyze**.

7. **env-config-2.2.jar** is detected as a runtime dependency. Click **Proceed**.

8. Click **Run automated fixes** to modernize the code. You can review the changes that have been made.

9. We want to create a new version of the jar file, so open the **pom.xml** file and update the **<version>** from 2.2 to 2.3. **Save** the file.

10. Right click on the **pom.xml** file and choose **Open in integrated termina**l.

11. Now build the jar file with this command: **mvn clean package**.

12. Now we will add this new jar file to the local maven repo.

    a. Go to the build location: **cd target**

    b. Add the jar file: **mvn install:install-file -Dfile=env-config-2.3.jar -DgroupId=com.acme.common -DartifactId=env-config -Dversion=2.3 -Dpackaging=jar**

    Note: When you copy this command to the VM make sure the whole command is executed, a line break may have been introduced.

    Normally this jar file would be published to a central repository and pulled from there, but for this lab we will just install it locally to give the same effect

4.4. Using the modernized common code

We will now update ModResorts to use the newly modernized common code and confirm that it works

1. Switch back to the **MOD-RESORTS-TX25** project in VSCode.

2. Open the **pom.xml** file and update the version of **env-config** to be **2.3**.

3. In the modernization panel click **Rebuild and Refresh** .

4. The issue in the external code is now gone.

5. In the left navigation open the **Liberty Dashboard**.

6. Right click on **modresorts** and choose **Start**.

7. The application is available at http://localhost:9080/resorts. There is also a bookmark for it in Firefox called **ModResorts Local**.

8. The application will open. Click the **Where to?** dropdown and select **Cork, Ireland**. Now you see that the page renders correctly! ModResorts has been fully modernized!! Well done.

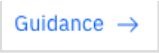
9. Right click on **modresorts** and choose **Stop**.

4.5. Modernization Guidance

We will review the guidance provided by AMA for the optimal sequence of modernizing your applications. The guidance is designed to let you build momentum by suggesting the easiest applications first, followed by the applications that have common code that will accelerate the modernization of later applications. It is designed to show you how to modernize the most applications as quickly as possible to accelerate the modernization of your whole Java estate.

1. Return to **AMA** and select the **TX2025** workspace.

2. Click on the **Assessment** tab.

3. Click on the **Guidance**  on the right-hand side of the screen.

4. Modernization guidance is divided into three sections based on the Complexity of the applications. Click on **Show more** in the first section (Simple applications).

5. This lists all the **Simple** applications. These applications can be deployed to the new runtime without any code changes and can be deployed in any order.

6. Now scroll down to the **Moderate** Applications. The first application listed is **gold-budgetting.ear**. Expand the accordion for it.

7. You can see that if you modernize this application, you will modernize **gold-calc-3.2.0.jar**, and that this jar can be reused in **gold-compliance.ear**. It is the only change that is required in **gold-compliance.ear** so it will reduce the complexity of that application to **Simple**.

8. Now open the accordion for **ReadinessAgent-1_2_war.ear**. You can see the **env-config-2.2.jar** is the only change that needs to be made. That is the change that you have just made to modernized ModResorts.

In our case we started with an application that was critical for the business, and we can see that by modernizing ModResort, we have fully modernized the ReadinessAgent (once we update it to use the new jar). However had we followed the guidance we would have been able to quickly update the ReadinessAgent first, test it and then move on to the more complicated case of ModResorts. In this simplified lab the sequence of modernization makes little difference, however in a typically customer estate with hundreds of applications and thousands of items of common code it is recommended that you follow the guidance to allow you see return on your modernization efforts quickly, while building towards the full modernization of your whole estate.

5. Extra Credit (Optional)

In this optional section we will return to the ACME capability and show how a configuration for the whole of ACME can be created in IBM Cloud.

5.5. Creating an IBM Cloud account

You can skip this step if you already have an IBM Cloud account.

1. Open IBM Cloud: https://www.ibm.com/cloud

2. Choose the option to **Sign up** and complete the steps necessary to create an account.

5.2. Creating an IBM Cloud API key

The API key will be used in the next step to create the ACME configuration in IBM Cloud

1. Log into IBM Cloud
2. Click **Manage > Access (IAM)**
3. In the left navigation click on **API Keys**
4. Click on **Create**
5. Enter a value for the name and click **Create**
6. In the dialog that pops up click **Copy**

5.3. Creating a configuration in IBM Cloud

We will use AMA to create a configuration in IBM Cloud for the all the components of ACME

1. Open Firefox. **AMA** should be the default page that opens. If not click on **AMA** in the bookmark bar.
2. Click on the **TX2025** workspace
3. Click in the **expander arrow** in the top left of the screen. Click on **Connected groups** to expand the list of groups.
4. Click on **Connected_Group_A(6).** This highlights the ACME group.
5. Click on the **Assessment** tab
6. Select Enterprise Application Service (EASeJ) as target destination
7. On the right-hang side click on **Group Migration**
8. The migration plan for the group opens. It contains all the applications, databases and messaging queues. Click through the entries in the left navigation to see the details of each element.
9. Now we will create this configuration in IBM Cloud. Click **Start in IBM Cloud**
10. Enter a **project name** and copy in the **API Key** you created in the previous section, then click **Next**
11. Select a **region** and a resource **group,** then click **Next**
12. Review the information then click **Next**

13. When the configuration has been created in IBM Cloud you will see a notification indicating that the project has been created. Click **View your project**.

14. IBM Cloud will open (you may need to log in) and it will show the project that has just been created. There will be a single row, click the **accordion icon to expand it**.

15. You will see eight rows in total

    a. One row for each application

    b. One row for the database

    c. One row for the queue manager. The queue manager contains the three queues that the ACME group require. These are placed in a single queue manager to reduce cost and footprint.

A configuration has been created in IBM Cloud for the ACME group. When these elements are deployed, they will be configured to connect to each other. Some manually work is still required to add sensitive credentials. This configuration will significantly accelerate the deployment of the ACME capability into IBM Cloud using EASeJ.

6. Getting help and troubleshooting

This section provides information about getting help with your lab and some common troubleshooting topics.

6.3. Reach out to the Lab instructor

The Lab instructors are available to help with any issues that you have and also to answer any questions about AMA and the AMA developer tools.

6.4. Common troubleshooting tips

The lab machine becomes unresponsive

In some cases, the connection can be lost. Close the window and open the VM Console again.

AMA is unavailable in the web browser

If for some reason AMA is unavailable in the web browser then run the start script again.

You have closed the modernization panel and need to reopen it

1. In the VSCode Explorer **right click** on any file

2. At the bottom of the pop-up menu select **Modernize to Liberty**

Maven is unable to find env-config dependency

Make sure that you have executed the command to add the updated env-config.jar to the local maven repo. See section 6.3.

Maven has cached the fact it can’t find env-config dependency

In some cases, maven may try to pull a version of env-config.jar that is not available at that time. When the jar is added later maven may still think the jar is unavailable. To resolve this issue, delete local reference to env-config.jar by

1. Running this command: **rm -Rf /home/admin/.m2/repository/com/acme/common/env-config**
2. Adding the env-config.jar to the local maven repo by executing the steps in Section 6.3
3. Building the ModResorts application again

3. Lab Reference Materials

Download AMA: https://www.ibm.com/account/reg/us-en/subscribe?formid=urx-53705

AMA Documentation: https://www.ibm.com/docs/en/ama

Application Modernization Playbook: https://ibm.github.io/app-mod-journey/tree/index.html

AMA Developer Tools

Eclipse Marketplace

Visual Studio Code Marketplace

Documentation

Get entitlement via Enterprise Application runtimes: https://www.ibm.com/docs/en/ear?topic=overview-whats-new-in-enterprise-application-runtimes