

# **Micro Focus and AWS COBOL CI/CD Pipeline Configuration Guide**

## Table of Contents

1	Introduction .....	4
1.1	Products Overview .....	4
1.2	Documentation .....	5
1.3	CI/CD pipeline description.....	6
1.4	Configuration samples .....	8
1.5	Feedback .....	8
2	Development environment .....	9
2.1	DEV – MF EDz configuration for Bankdemo.....	9
2.2	DEV – MF EDz configuration for Bankdemo unit test server .....	14
2.3	DEV – MF EDz BankDemo online unit test .....	20
3	SOURCE stage .....	24
3.1	SOURCE – MF EDz and AWS CodeCommit integration.....	24
3.2	SOURCE – MF EDz to AWS CodeCommit code push .....	31
4	BUILD stage .....	34
4.1	BUILD – Amazon ECR docker image with Micro Focus build tools.....	34
4.2	BUILD – AWS CodeBuild configuration .....	42
4.3	BUILD – AWS CodePipeline configuration .....	49
5	DEPLOY to TEST stage .....	53
5.1	DEPLOY to TEST – MF ETS configuration for CodeDeploy .....	53
5.2	DEPLOY to TEST – AWS CodeDeploy configuration .....	60
5.3	DEPLOY to TEST – AWS CodePipeline configuration .....	65
6	TEST stage.....	68
6.1	TEST – MF ETS configuration for automated tests .....	68
6.2	TEST – MF ETS configuration for SSM .....	72
6.3	TEST – AWS Lambda configuration.....	75
6.4	TEST – AWS CodePipeline configuration .....	78
7	DEPLOY to PROD stage .....	81
7.1	DEPLOY to PROD – AWS Lambda configuration .....	81
7.2	DEPLOY to PROD – AWS CodePipeline configuration .....	84
8	Appendix .....	87
8.1	Configure instance for new Administrator random password.....	87
8.2	Powershell script to compile Bankdemo on EDz.....	87

## Revision History

Date	Author(s)	Version	Comment
<b>2020-03-31</b>	Phil de Valence (Amazon Web Services) Mathias Mezger (Micro Focus) Gary Evans (Micro Focus)	V1.0	First baseline version

# 1 Introduction

---

## 1.1 Products Overview

### **Micro Focus Enterprise Developer for z Systems**

Enterprise Developer for z Systems (EDz) supports IBM COBOL, IBM PL/I, IBM Assembler, IBM CICS, IBM IMS, IBM JCL, IBM DB2, IBM z/OS file formats and common batch utilities, including SORT. This means that you can develop and maintain the core mainframe online and batch applications under Enterprise Developer.

EDz gives customers the choice to develop directly on the mainframe or under Windows. Mainframe applications can be developed, maintained and modernized regardless of where they will be deployed, either back onto the mainframe or onto an alternative server environment. Support is provided for both the Visual Studio and Eclipse-based IDEs and for all the development and test tools for every target environment currently supported by Micro Focus - including z/Linux, AIX and x86 environments.

### **Micro Focus Enterprise Test Server**

Enterprise Test Server (ETS) is a test execution environment that allows you to test mainframe applications on a lower cost Windows platform. You can use variables or the tilde syntax to relieve the mainframe test bottleneck by allowing you to perform a substantial part of your application testing on Windows prior to moving the application back to the mainframe for final pre-production testing and deployment.

ETS supports IBM COBOL, High Level Assembler, CICS, IMS TM and DB, JCL, DB2, and z/OS file formats. It also supports common batch utilities such as SORT. This means that applications running under ETS behave just as they would on the mainframe, so you can perform a wide variety of pre-production testing activities on low-cost hardware rather than on the mainframe.

### **AWS CodeCommit**

AWS CodeCommit is a fully-managed source control service that hosts secure Git-based repositories. It makes it easy for teams to collaborate on code in a secure and highly scalable ecosystem. CodeCommit eliminates the need to operate your own source control system or worry about scaling its infrastructure. You can use CodeCommit to securely store anything from source code to binaries, and it works seamlessly with your existing Git tools.

AWS CodeCommit eliminates the need to host, maintain, back up, and scale your own source control servers. The service automatically scales to meet the growing needs of your project. It has a highly scalable, redundant, and durable architecture. The service is designed to keep your repositories highly available and accessible.

### **AWS CodePipeline**

AWS CodePipeline automates your software release process, allowing you to rapidly release new features to your users. With CodePipeline, you can quickly iterate on feedback and get new features to your users faster.

Automating your build, test, and release process allows you to quickly and easily test each code change and catch bugs while they are small and simple to fix. You can ensure the quality of your application or infrastructure code by running each change through your staging and release process.

With AWS CodePipeline, you can immediately begin to model your software release process. There are no servers to provision or set up. CodePipeline is a fully managed continuous delivery service that connects to your existing tools and systems.

## AWS CodeBuild

AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy. With CodeBuild, you don't need to provision, manage, and scale your own build servers. CodeBuild scales continuously and processes multiple builds concurrently, so your builds are not left waiting in a queue. You can get started quickly by using prepackaged build environments, or you can create custom build environments that use your own build tools. With CodeBuild, you are charged by the minute for the compute resources you use.

## AWS CodeDeploy

AWS CodeDeploy is a fully managed deployment service that automates software deployments to a variety of compute services such as Amazon EC2, AWS Fargate, AWS Lambda, and your on-premises servers. AWS CodeDeploy makes it easier for you to rapidly release new features, helps you avoid downtime during application deployment, and handles the complexity of updating your applications. You can use AWS CodeDeploy to automate software deployments, eliminating the need for error-prone manual operations. The service scales to match your deployment needs.

## AWS Lambda

AWS Lambda lets you run code without provisioning or managing servers (serverless). You pay only for the compute time you consume. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.

## 1.2 Documentation

Micro Focus Enterprise Developer 5.0 for Eclipse documentation:

<https://www.microfocus.com/documentation/enterprise-developer/ed50/ED-Eclipse/GUID-8D6B7358-AC35-4DAF-A445-607D8D97EBB2.html>

Micro Focus Enterprise Test Server 5.0 documentation:

<https://www.microfocus.com/documentation/enterprise-developer/ed50/ETS-help/GUID-ECA56693-D9FE-4590-8798-133257BFEBE7.html>

AWS CodeCommit documentation:

<https://docs.aws.amazon.com/codecommit/latest/userguide/welcome.html>

AWS CodeBuild documentation:

<https://docs.aws.amazon.com/codebuild/latest/userguide/welcome.html>

AWS CodeDeploy documentation:

<https://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html>

AWS CodePipeline documentation:

<https://docs.aws.amazon.com/codepipeline/latest/userguide/welcome.html>

AWS Lambda documentation:

<https://docs.aws.amazon.com/lambda/latest/dg/welcome.html>

AWS Systems Manager documentation:

<https://docs.aws.amazon.com/systems-manager/latest/userguide/what-is-systems-manager.html>

## 1.3 CI/CD pipeline description

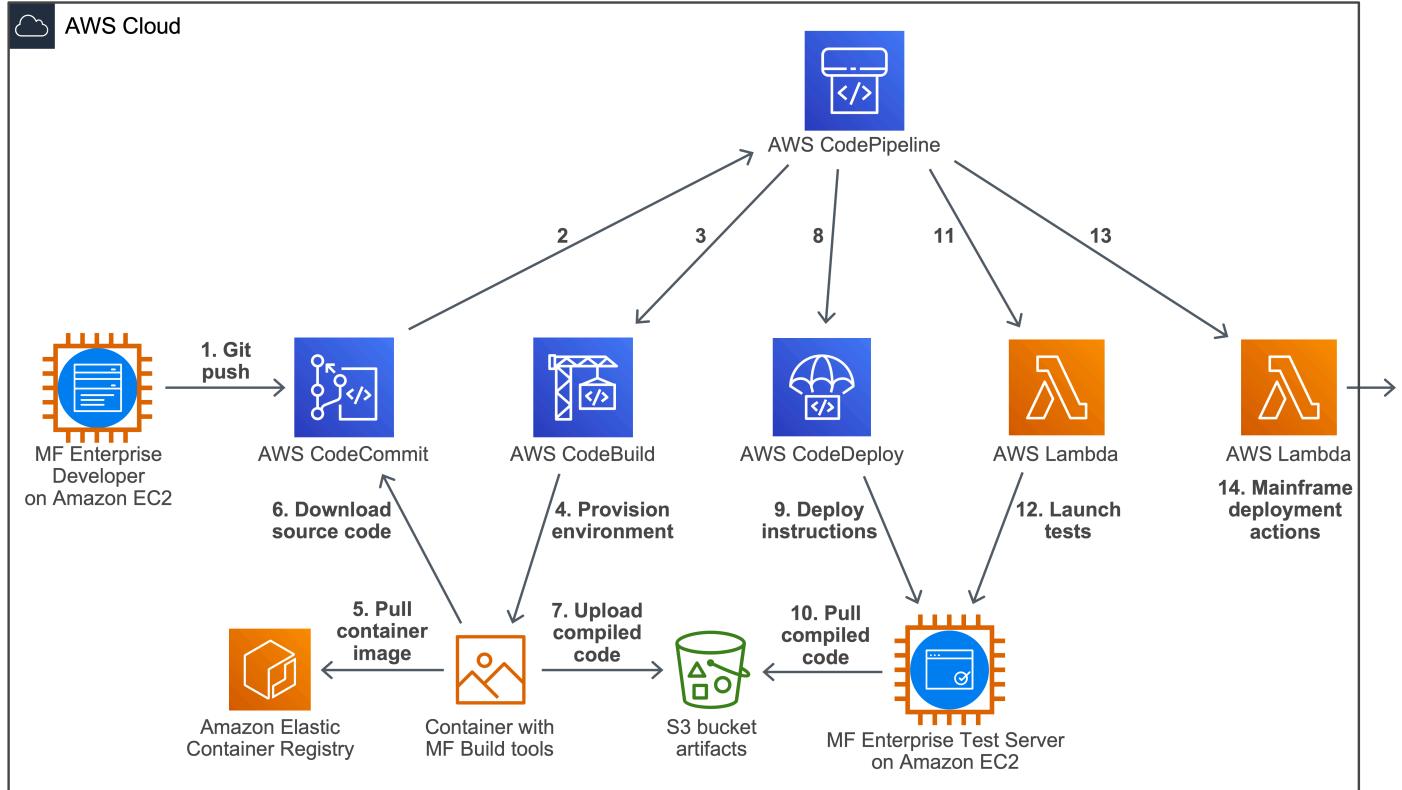
This document describes how to configure a COBOL Continuous Integration / Continuous Delivery (CI/CD) pipeline using Micro Focus (MF) and Amazon Web Services (AWS) components.

You can learn more about the pipeline incremental approach and components in this blog post:

<https://aws.amazon.com/blogs/industries/enable-agile-mainframe-development-test-and-ci-cd-with-aws-and-micro-focus/>

This pipeline is modular. This means you can swap some of the tools and you can decide to configure one stage but not the others.

The pipeline includes Micro Focus Enterprise Developer (MF EDz), Micro Focus Enterprise Test Server (MF ETS).



The CI/CD pipeline shown above executes the following steps:

1. A developer makes changes to the source code and commits the changes to the local Git repository. These source code changes are pushed to the upstream repository hosted by AWS CodeCommit.
2. The source code changes in CodeCommit trigger an AWS CloudWatch event which starts the pipeline in AWS CodePipeline.
3. The pipeline calls AWS CodeBuild in order to start the build phase.
4. AWS CodeBuild provisions a build environment in a container with Micro Focus Enterprise Developer Build tools.
5. This container is based on a container image pulled from Amazon Container Registry.
6. Once the container is provisioned, the build environments download the source code from AWS CodeCommit. The source code is compiled and linked.
7. Then the compiled code is uploaded in an Amazon S3 buckets which stores the generated build artifacts.
8. The build phase being complete, the pipeline calls AWS CodeDeploy as part of the test phase.
9. AWS CodeDeploy send code deployment instructions to the CodeDeploy agent residing on the Amazon EC2 instance hosting Micro Focus Enterprise Test Server.
10. The AWS CodeDeploy agent pulls the compiled code from the S3 bucket with the artifacts and deploys it to the proper destination folders and restarts Micro Focus Enterprise Test Server.
11. Once the code is deployed, the pipeline calls AWS Lambda to start the tests.
12. AWS Lambda sends the test command to the Amazon EC2 test instance via Amazon Systems Manager (SSM). The test command triggers a batch test script on the test instance. The batch test script calls a Visual Basic script with can either trigger a Rumba or a UFT automation script. The test script executes test cases against the modified compiled code and Micro Focus Enterprise Test Server verifying the new code is operational. The result of the tests is sent back to the pipeline in AWS CodePipeline.

13. If tests are successful, the pipeline calls AWS Lambda in order to send the source code back to the mainframe Source Code Management (SCM) system.
14. The AWS Lambda function retrieves the code changes from AWS Code Commit, and processes the modified files for deploying to production.

## 1.4 Configuration samples

In order to facilitate the configuration of this pipeline, we have made available some reusable sample configuration files on GitHub: <https://github.com/aws-samples/mainframe-cobol-cicd-pipeline-aws-microfocus>

For example, you will find the Lambda functions and the CodeBuild or CodeDeploy configuration files on GitHub.

## 1.5 Feedback

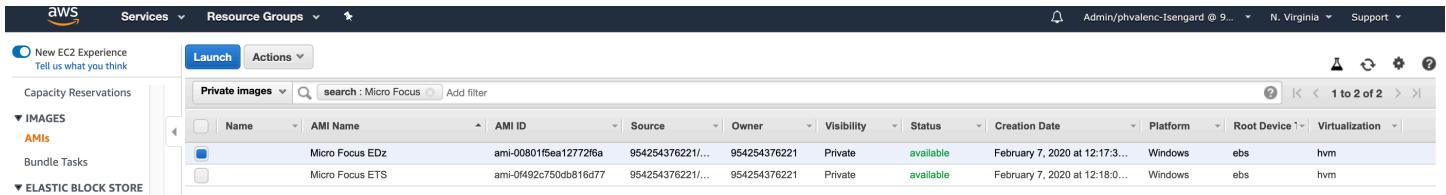
We are glad to improve this pipeline, the configuration samples, and this documentation. If you have comments, suggestions, or challenges, feel free to let us know and we will be glad to help. You can contact us via your AWS representative, your Micro Focus representative, or by opening an issue on GitHub: <https://github.com/aws-samples/mainframe-cobol-cicd-pipeline-aws-microfocus/issues>

# 2 Development environment

## 2.1 DEV – MF EDz configuration for Bankdemo

For getting access to Micro Focus software, please contact your Micro Focus representative or contact Micro Focus following this link: <https://www.microfocus.com/en-us/contact/contactme>

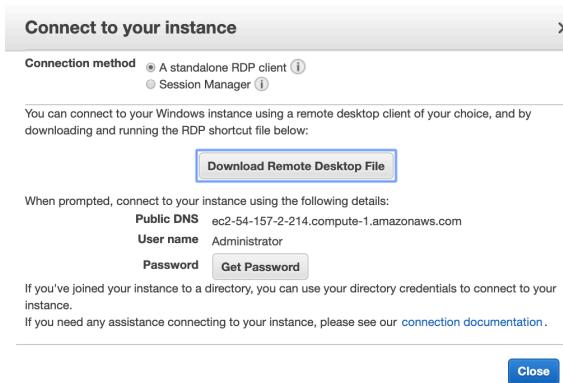
First you need to retrieve an AMI with Micro Focus Enterprise Developer (EDz) or deploy the EDz software on an EC2 instance.



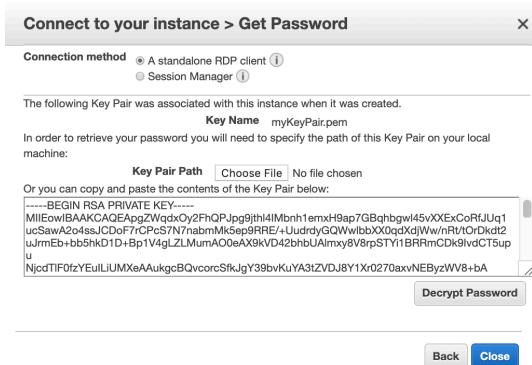
Name	AMI Name	AMI ID	Source	Owner	Visibility	Status	Creation Date	Platform	Root Device	Virtualization
Micro Focus EDz	ami-00801f5ea12772f6a	954254376221...	954254376221	Private	available	February 7, 2020 at 12:17:3...	Windows	ebs	hvm	
Micro Focus ETS	ami-0f492c750db816d77	954254376221...	954254376221	Private	available	February 7, 2020 at 12:18:0...	Windows	ebs	hvm	

If you retrieve Micro Focus software from an AMI, launch EC2 instance from Micro Focus EDz AMI.

Once started, choose to Connect to the instance.



Download Remote Desktop File and Get Password



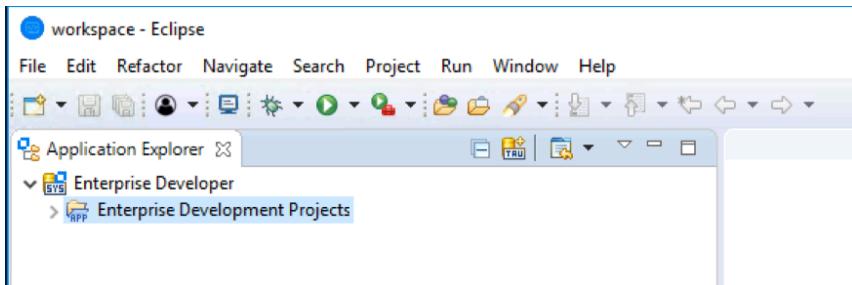
Decrypt password.

Open RDP file and connect to instance with Administrator username and decrypted password.

Start Enterprise Developer for Eclipse



Go to Application Explorer view

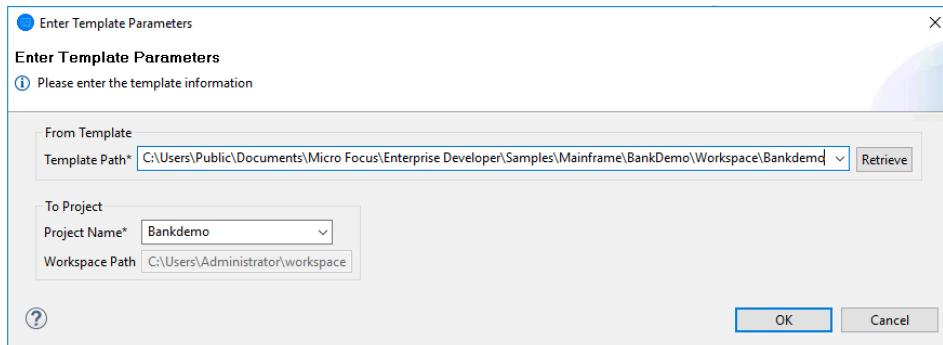


Right-click Enterprise Development Projects, which is the node for the standard application.

Click New Project from Template.

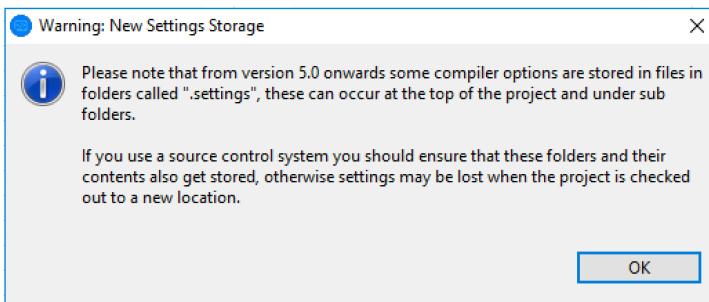
You can find more details for help at this link:

<https://www.microfocus.com/documentation/enterprise-developer/ed50/ED-Eclipse/GUID-5A27996B-F03D-471A-ACDF-36E680E5939E.html>

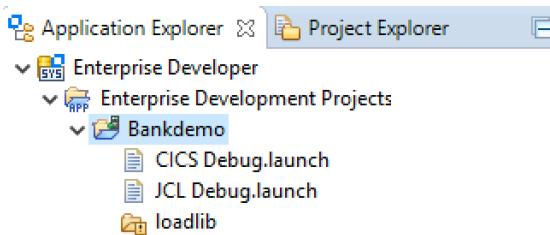


Browse to the C:\Users\Public\Documents\Micro Focus\Enterprise Developer\Samples>Mainframe\BankDemo\Workspace\Bankdemo and click Select Folder.

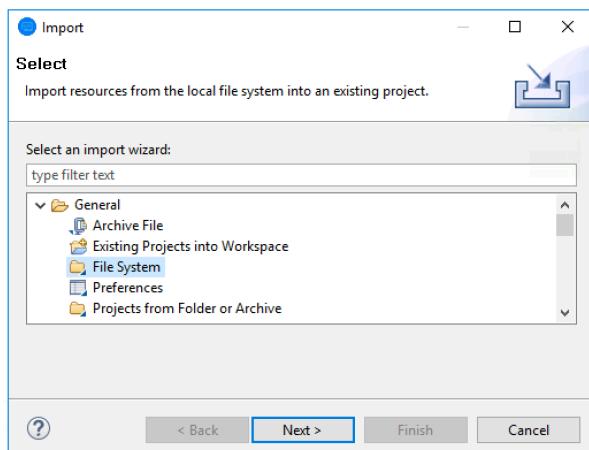
Type Bankdemo in the project name field and click OK.



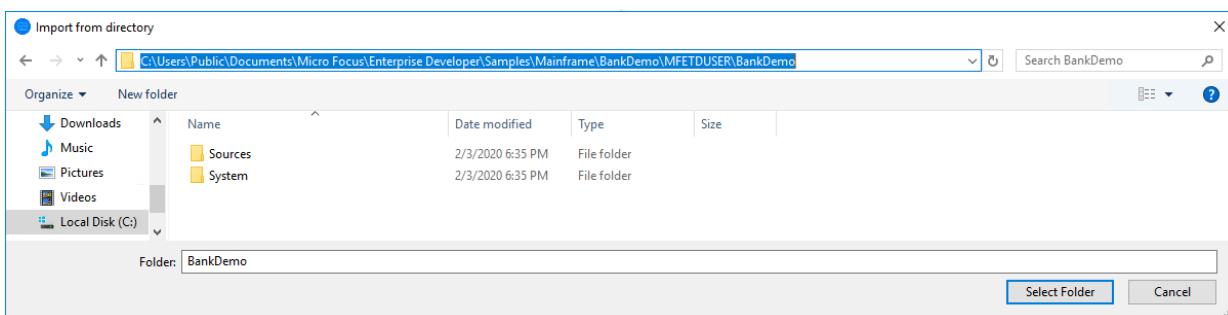
This creates a copy of the Bankdemo project in your Eclipse workspace and adds the project to the application in the Application Explorer view. If the Bankdemo project entry is not displayed in the tree view, refresh the Enterprise Development Projects entry.



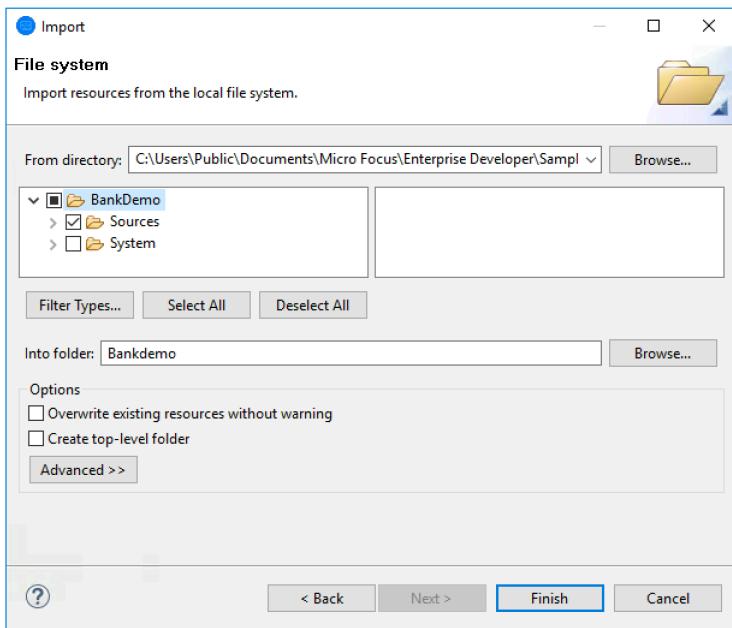
Add the demo source files, in Application Explorer view, right-click the Bankdemo project, and click Import...



Next



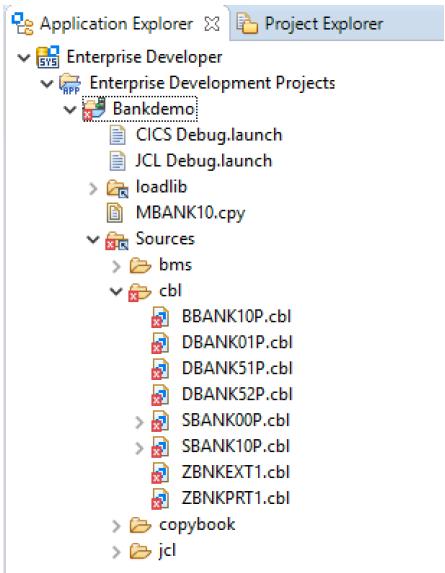
Select folder C:\Users\Public\Documents\Micro Focus\Enterprise Developer\Samples>Mainframe\BankDemo\MFETDUSER\BankDemo



Select Source subdirectory.

Click Finish.

This adds the folders storing the source files to your project as imported resources:



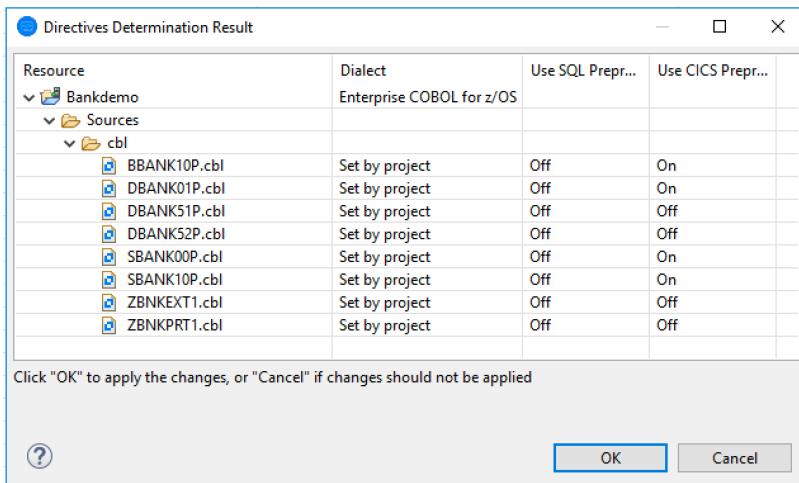
By default, Eclipse is set to build projects automatically (see Project > Build Automatically), so it immediately compiles the files you have just added.

Check the Console and Problems views and see that some of the programs failed to compile. In Application Explorer view, you can select a program and check the error and warning count property in the Properties view to indicate that there are compilation problems.

The errors are due to the correct copybook paths not being specified in the project properties.

Right-click Bankdemo and click Determine Directives.

The IDE performs a scan of the files and shows a report of what directives for dialect and for CICS must be set on the programs in your project in order for them to compile cleanly.



Click OK to set the directives.

If you are prompted to delete some user files, choose No.

Setting the missing Compiler directives triggers a full rebuild of the project. There are still some errors in the COBOL sources due to the fact that the project cannot resolve the paths to the copybook files.

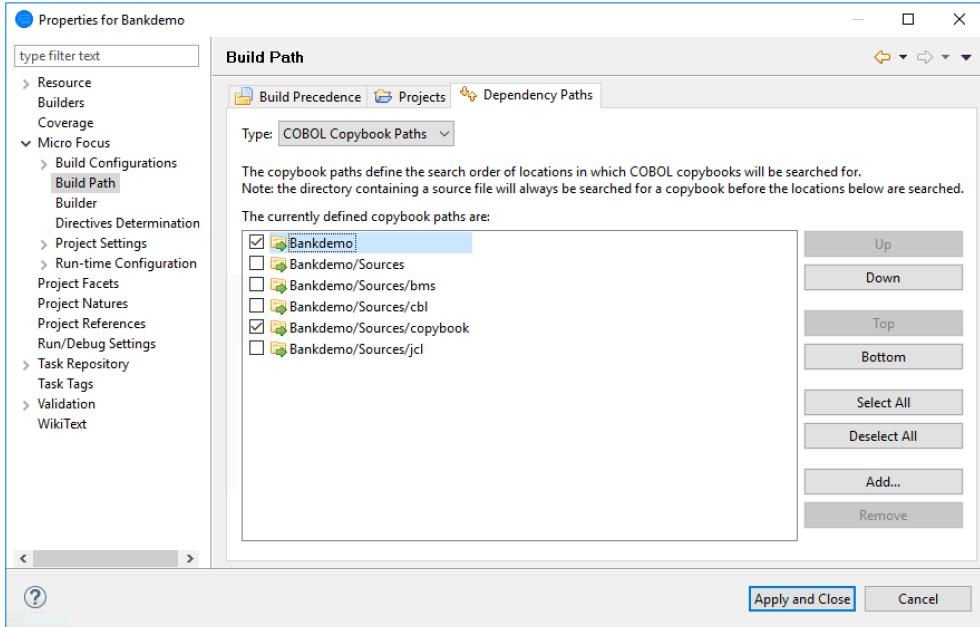
Then we specify the copybook paths for the project.

In Application Explorer view, right-click the Bankdemo project, and click Properties.

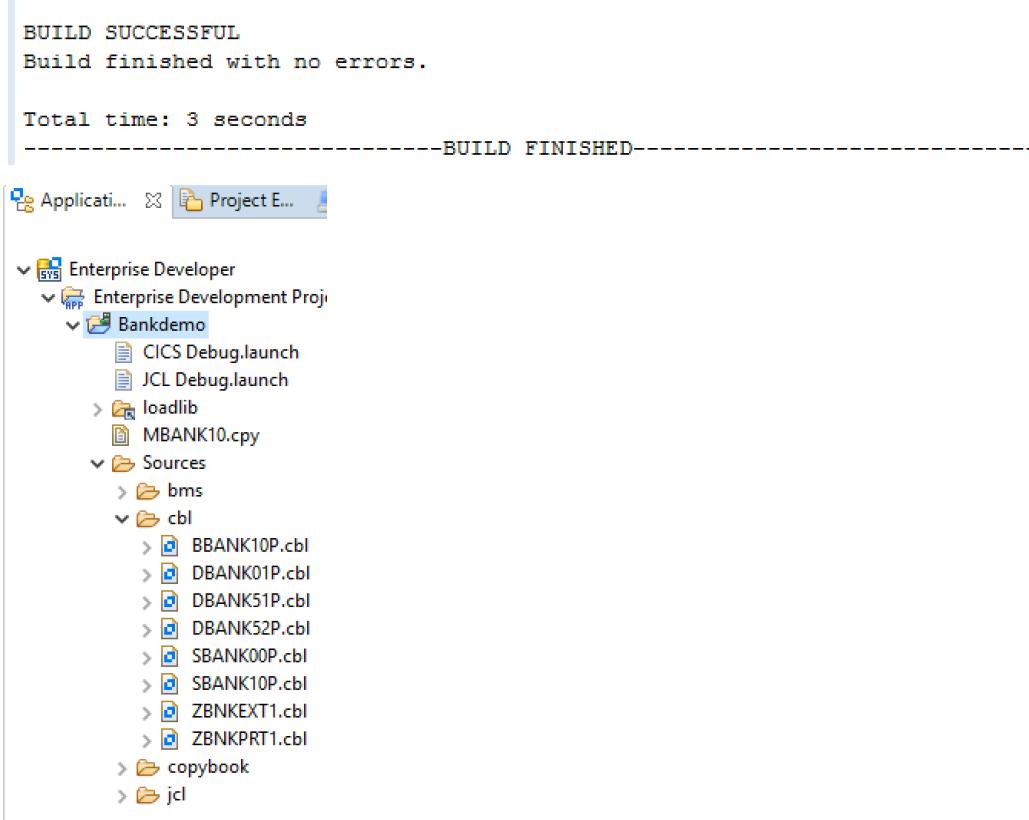
Expand Micro Focus, and then click Build Path.

Click the Dependency Paths tab and ensure Type is set to COBOL Copybook Paths.

Check the Bankdemo/Sources/copybook entry in the list, and then click Apply and Close.



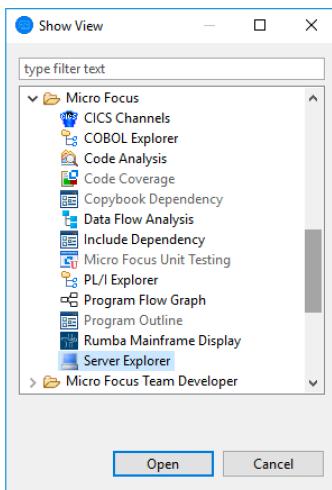
Eclipse rebuilds the project and now all COBOL programs should compile cleanly.



## 2.2 DEV – MF EDz configuration for Bankdemo unit test server

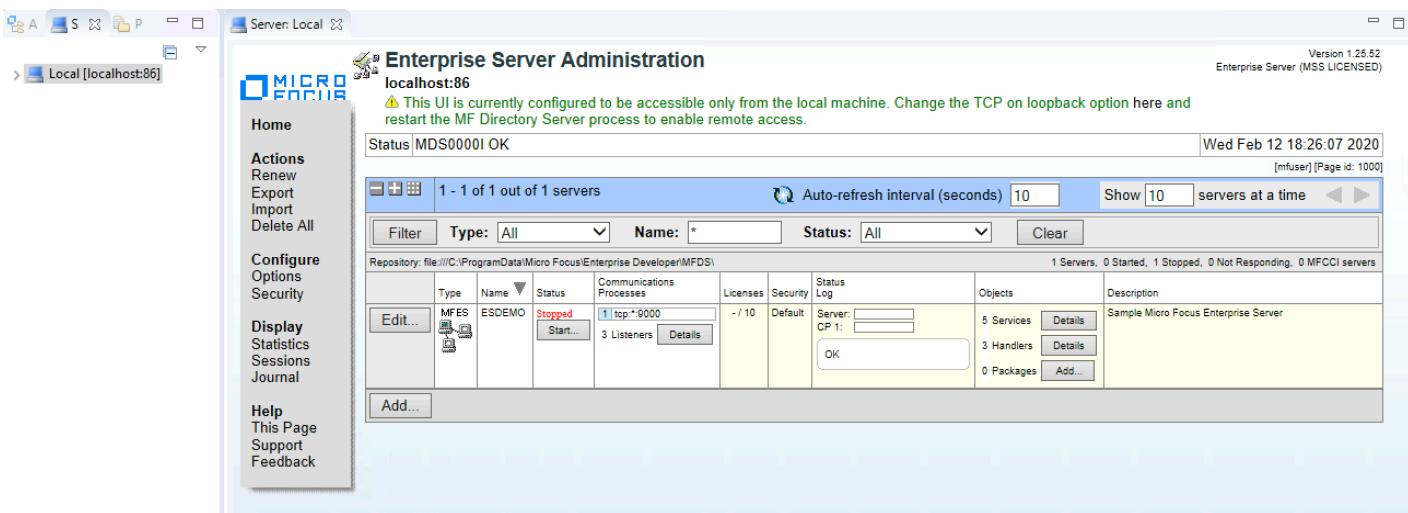
You can find more details for help at this link:

Click the Server Explorer tab. If the tab is not visible, click Window > Show View > Other. Select Micro Focus > Server Explorer and then click Open.



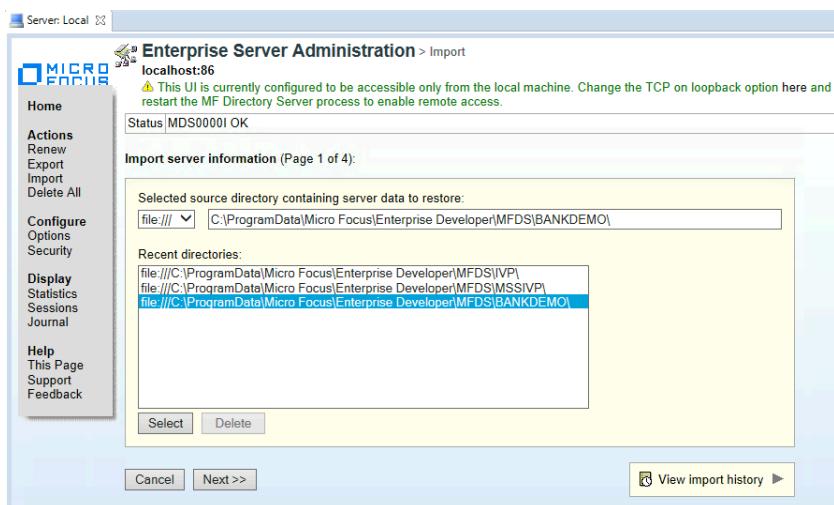
First, we import the Bankdemo server.

In Server Explorer, right-click Local [localhost:86] and click Open Administration Page.



Click Import in the left upper corner of Enterprise Server Administration.

On the Import server information page and under Recent directories click the directory for the BANKDEMO server.



This adds the path to the Selected source directory containing server data to restore field.

Click Next 3 times and then OK to import the BankDemo server (keep the BankDemo server in 32 bit).

The system returns to the main Enterprise Server Administration page.

Type	Name	Status	Processes	Licenses	Security	Status Log	Objects	Description
MFES (MSS)	BANKDEMO	Stopped	1 top.* 3 Listeners	- / 10	Default	Server: CP 1: OK 38 seconds in "Stopped" state since 16:00:08 02/25/20	4 Services 4 Handlers 0 Packages	Sample server for BANKDEMO application
MFES (MSS)	ESDEMO	Not Responding	1 top:172.31.33.236*9000 (172.31.33.236) 3 Listeners	- / 10	Default	Server: CP 1: MDS3801I Server started successfully 19:22:38 02/12/20 11 days 18 hours 10 minutes 55 seconds in "Not Responding" state since 21:58:47 02/13/20 Started externally using ES ID "mf_mds" under system ID "Administrator"	5 Services 3 Handlers 0 Packages	Sample Micro Focus Enterprise Server

You can see the Bankdemo server appears in the list of servers.

In front of the BANKDEMO server, select Edit...

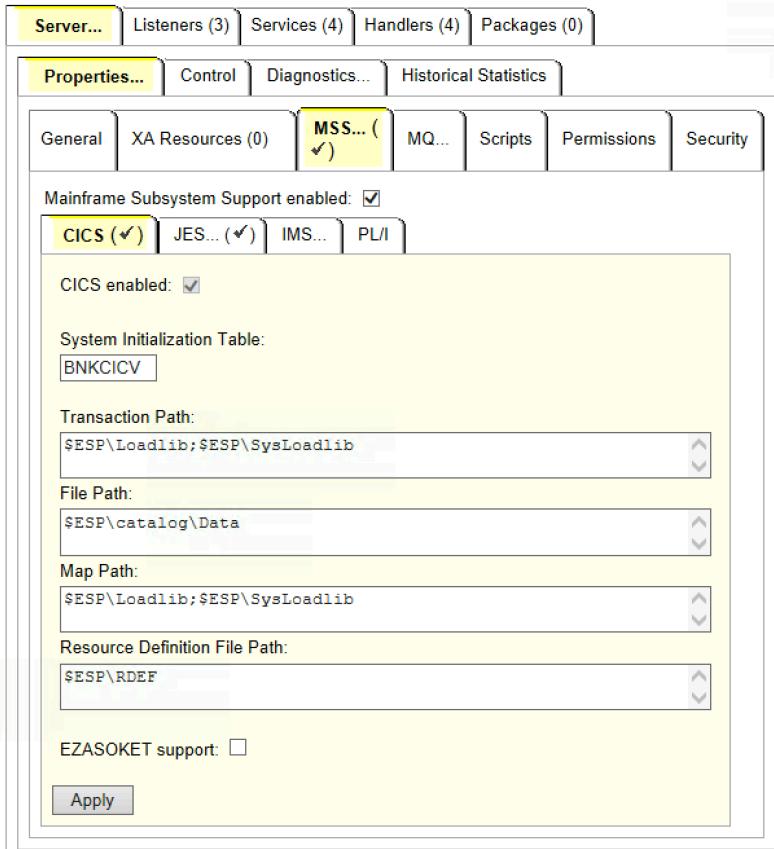
The screenshot shows the Micro Focus Enterprise Server Administration interface. The top navigation bar includes 'Enterprise Server Administration > BANKDEMO > Edit' and 'localhost:86'. A status message at the top right says, 'This UI is currently configured to be accessible only from the local machine. Change the TCP on loopback option here and restart the MF Directory Server process to enable remote access.' The left sidebar contains links for Home, Actions (Renew, Export, Import, Delete All), Configure (Options, Security), Display (Statistics, Sessions, Journal), and Help (This Page, Support, Feedback). The main content area displays the configuration for 'Server BANKDEMO [ Started ✓ ]'. It has tabs for Server..., Listeners (3), Services (4), Handlers (4), Packages (0), Properties..., Control, Diagnostics..., and Historical Statistics. The General tab is selected. The 'Name' field is set to 'BANKDEMO'. The 'System Directory' is '\$ESP\LOGS'. Under 'Startup Options', settings include Shared Memory Pages (512), Service Execution Processes (2), Shared Memory Cushion (32), Requested Licenses (10), Show Local Console (unchecked), Allow Dynamic Debugging (checked), Start on System Start (unchecked), 64-Bit Working Mode (unchecked), Purge Old Logs (unchecked), and Console Log Size (K) (0). The 'Windows Monitoring and Management' section includes 'Allow Performance Monitoring' (unchecked) and 'Event Logging' checkboxes for Informational, Warning, Error, and Severe levels. The 'Configuration Information' section shows environment variables: [ES-Environment], ESP=C:\Users\Public\Documents\Micro Focus\Enterprise Developer\Samples>Mainframe\BankDemo\MFETDUSER\BankDemo\System, and MF\_CHARSET=E. The 'Description' section contains 'Sample server for BANKDEMO application'. At the bottom are buttons for Cancel, OK, Apply, Export..., Copy..., Delete, and Validate.

Under Configuration Information, update the ESP variable path to the actual location on EDz:

```
ESP=C:\Users\Public\Documents\Micro Focus\Enterprise
Developer\Samples>Mainframe\BankDemo\MFETDUSER\BankDemo\System
```

Click OK to save.

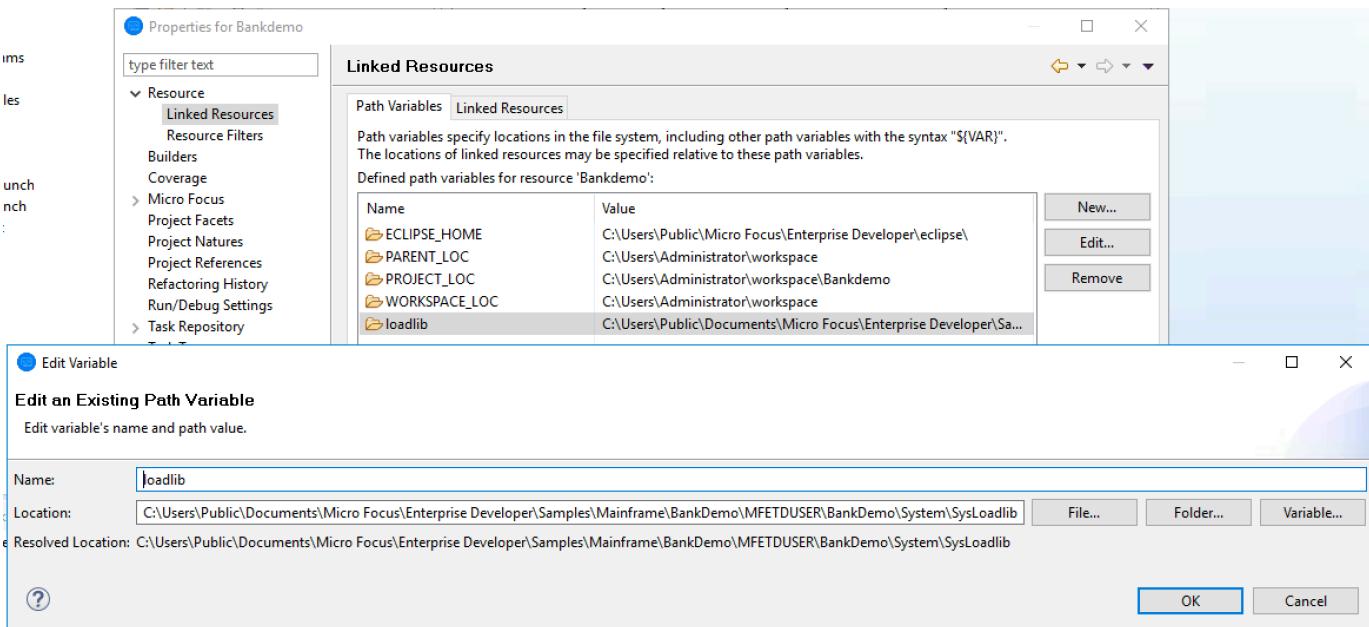
Make sure the CICS paths for SysLoadlib configuration match the file system.



C:\Users\Public\Documents\Micro Focus\Enterprise Developer\Samples>Mainframe\BankDemo\MFETDUSER\BankDemo\System			
Name	Date modified	Type	Size
catalog	2/12/2020 7:47 PM	File folder	
DATA	2/20/2020 11:09 PM	File folder	
Logs	2/25/2020 2:56 PM	File folder	
RDEF	2/3/2020 6:35 PM	File folder	
SysLoadlib	2/15/2020 2:12 AM	File folder	

Be careful with the SysLoadlib case. You may have to adjust the folder name for proper upper-case letters.

For the Bankdemo project, under Properties -> Resource -> Linked Resources -> Path Variables, Edit the loadlib variable to match the BANKDEMO server loadlib folder in C:\Users\Public\Documents\Micro Focus\Enterprise Developer\Samples>Mainframe\BankDemo\MFETDUSER\BankDemo\System\sysloadlib

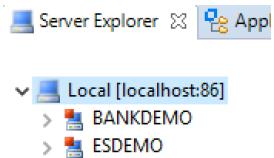


Then choose Apply and Close.

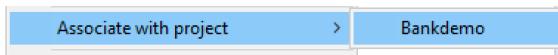
Clean the Bankdemo project to rebuild.

Then we associate the Bankdemo Enterprise Server with the application project.

In Server Explorer, right-click Local, and click Refresh to show the BankDemo server. If you get a Directory Server offline message appears, restart EDz.



Right-click the BANKDEMO server, point to Associate with project, then select Bankdemo.



Configure EDz to start the associated BankDemo server automatically as follows. Click Window > Preferences.

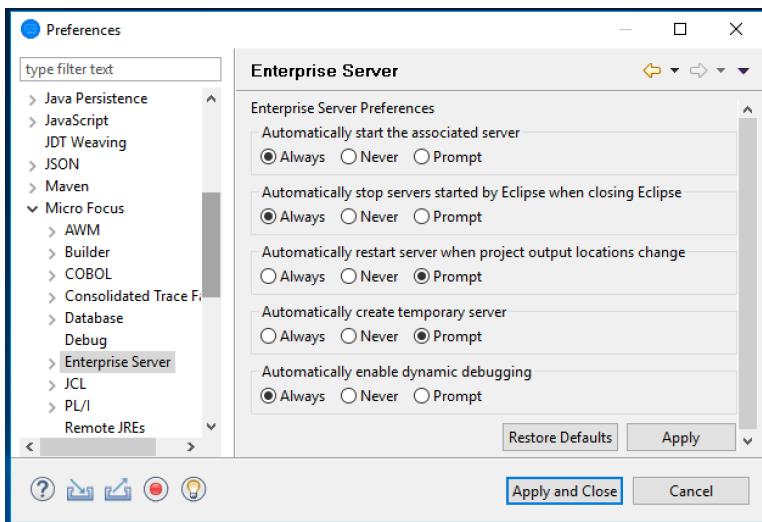
Expand Micro Focus, and click Enterprise Server.

Set the following options on this page to Always in order to enable the IDE to start or stop the associated server, and to enable dynamic debugging, for when it is not enabled in the server:

Automatically start the associated server - this ensures the IDE will start the server if it is not running when you execute the application.

Automatically stop servers started by Eclipse when closing Eclipse - this enables the IDE to stop servers when you close Eclipse.

Automatically enable dynamic debugging - this ensures the IDE will check whether the server has dynamic debugging enabled and, if it is not, will enable it when you start debugging.



Click Apply and Close.

In the Server Explorer window, right-click the Bankdemo server and then click Start.

Wait until the server has started. In the list of servers in Server Explorer, BANKDEMO still has a red square next to it. This is a refresh delay.

If any server start problem you can check the logs doing a right-click on BANKDEMO server then Show Console Log or in C:\Users\Public\Documents\Micro Focus\Enterprise Developer\Samples>Mainframe\BankDemo\MFETDUSER\BankDemo\System\Logs\console.txt.

```

AWS Explorer Console Problems Tasks Properties Table Results Filter Definitions Micro Focus Unit Testing Code Coverage
Enterprise Server Log: Local/BANKDEMO
200212 19472147 3568 BANKDEMO CASSI1600I SEP initialization completed successfully 19:47:21
200212 19472149 3604 BANKDEMO CASCL0020I Printer L860 initialized successfully 19:47:21
200212 19472163 5156 BANKDEMO CASBJ0005I Batch initiator started for job classes "A" 19:47:21
200212 19472169 3120 BANKDEMO CASBJ0021I Batch printer started for print output classes "A" 19:47:21
200212 19472169 2124 BANKDEMO CASBJ0005I Batch initiator started for job classes "AB" 19:47:21
200212 19472169 4516 BANKDEMO CASSI1600I SEP initialization completed successfully 19:47:21
200212 19472170 3120 BANKDEMO JES0000029I Locating batch printer: MTO-HTML 19:47:21
200212 19472175 4516 BANKDEMO CASSI5020I PLTPI Phase 2 List(PI) Processing Started 19:47:21
200212 19472175 3120 BANKDEMO JES0000029I Available batch printer: Microsoft XPS Document Writer 19:47:21
200212 19472177 3120 BANKDEMO JES0000029I Available batch printer: Microsoft Print to PDF 19:47:21
200212 19472177 3120 BANKDEMO JES0000025I Printer not found: MTO-HTML 19:47:21
200212 19472177 3120 BANKDEMO JES0000024I Batch Printer: MTO-HTML 19:47:21
200212 19472177 3120 BANKDEMO JES0000026I Exit Module: ZBNKPRNX 19:47:21
200212 19472178 4516 BANKDEMO CASOP00001 From (,,FIP2) UBNKPLT2 Complete 19:47:21
200212 19472178 4516 BANKDEMO CASSI5030I PLTPI Phase 2 List(PI) Processing Completed 19:47:21

```

## 2.3 DEV – MF EDz BankDemo online unit test

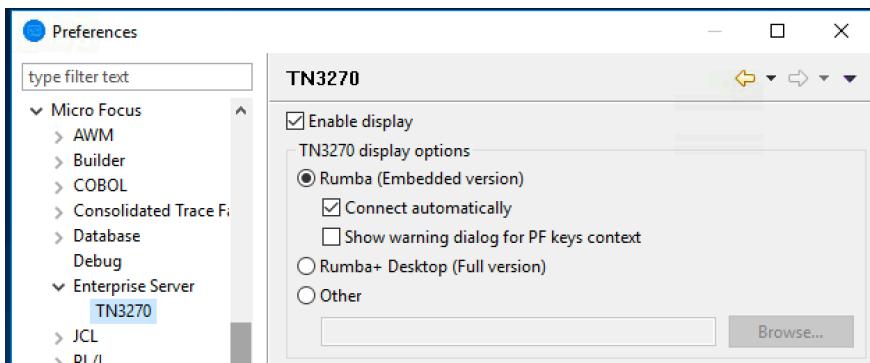
You can find more details for help at this link:

<https://www.microfocus.com/documentation/enterprise-developer/ed50/ED-Eclipse/index.html?t=GUID-A0221822-CF44-4698-ABD6-7F77F7A862A2.html>

In the EDz IDE, click Window > Preferences.

Expand Micro Focus > Enterprise Server, and click TN3270.

Ensure that Enable display, Rumba (Embedded) and Connect automatically are all selected.

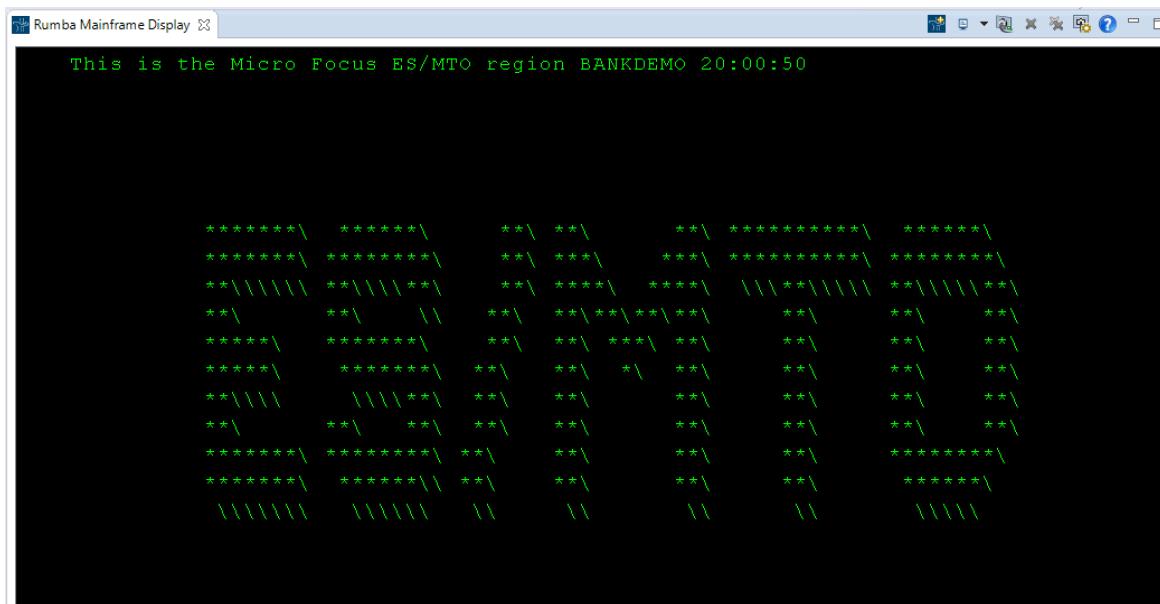


Click OK.

Right-click the BANKDEMO server in Server Explorer, and click Show TN3270 Display.

This opens the Rumba+ Desktop Mainframe Display view and automatically establishes a 3270 terminal connection to the BANKDEMO server. You can see the starting page of the ES/MTO region BANKDEMO.

You can drag the Rumba Mainframe Display view to reposition it.



Press Ctrl+Shift+Z to clear this window.

Type in the transaction id BANK and press Enter (Ctrl) to navigate to an application logon window.

Rumba Mainframe Display ☰ Server Local

Scrn: BANK10 Enterprise Developer Demonstration 13.Feb.2020  
Tran: BANK \*\*\*\*  
  
Welcome to Micro Focus' demonstration system based on a mythical bank.  
  
Please identify yourself to the system by entering your user id  
and password below and pressing Enter  
  
User id.....: b0001  
Password.....: █

Type your logon details and press Enter. A suitable User Id is b0001. You can type anything as a Password - the field must not be empty though.

Rumba Mainframe Display ☰ Server Local

Scrn: BANK20 Enterprise Developer Demonstration 13.Feb.2020  
Tran: BANK \*\*\*\*  
  
Welcome to the almost on-line banking system.  
  
Please select one of the following options:  
  
/\_ Display your account balances  
\_ Transfer funds between accounts  
\_ Update your contact information  
\_ Calculate the cost of a loan  
\_ Request printed statement(s)  
\_ Obtain more information

Type / against Display your account balances and press Enter to see the details for this customer.

Rumba Mainframe Display ☰ Server: Local

Scrn: BANK30 Enterprise Developer Demonstration 13.Feb.2020  
Tran: BANK \*\*\*\*\* 20:11:09

Account balance information for:  
B0001 - Fred Bloggs

Select account to see details or transactions

Det	Account No.	Account type	Current Balance	Service Charge	Last Statement	Txns
█	450061494	Savings	235.32	25.00	10.Jan.2006	
—	737590226	Checking	0.00	50.00	10.Jan.2006	
—	541374829	Investment	3.60	25.00	10.Jan.2006	*

You can explore this application further if you wish or press F3 to terminate it.

# 3 SOURCE stage

## 3.1 SOURCE – MF EDz and AWS CodeCommit integration

You can find more details for help at this link:

<https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-ide-ec.html>

In AWS IAM, create a new AWS user for integration

Add user

Set user details

User name\* CICD-User

Add another user

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

Access type\*  Programmatic access  
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

AWS Management Console access  
Enables a **password** that allows users to sign-in to the AWS Management Console.

Add user

Set permissions

Add user to group

Copy permissions from existing user

Attach existing policies directly

Create policy

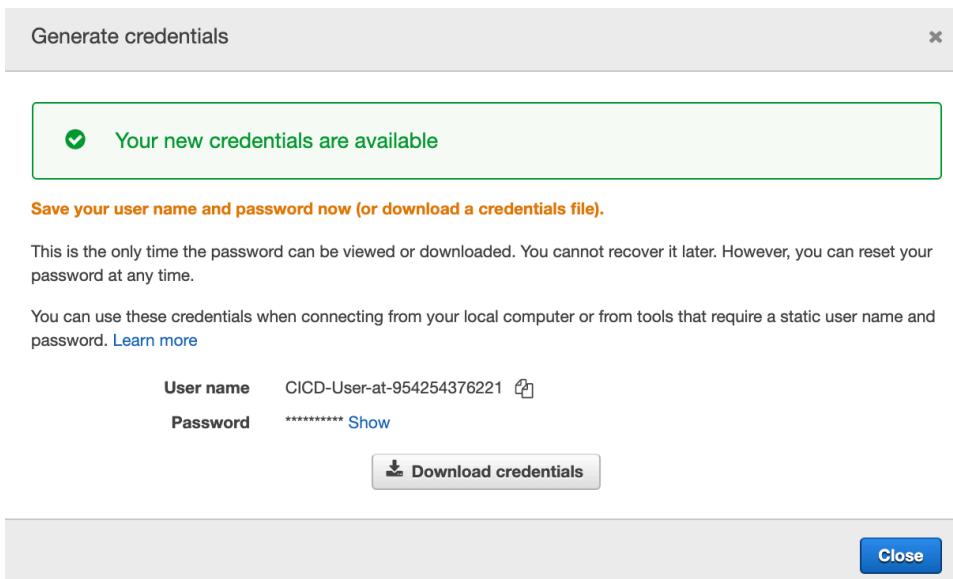
Filter policies ▾

Policy name	Type	Used as
<input type="checkbox"/> AWSCodeCommitFullAccess	AWS managed	None
<input checked="" type="checkbox"/> AWSCodeCommitPowerUser	AWS managed	None
<input type="checkbox"/> AWSCodeCommitReadOnly	AWS managed	None

Attach the AWSCodeCommitPowerUser policy and create User.

Download and save the Access Key and the Secret Access Key.

Select the User, then Under Security Credentials, go to HTTPS Git credentials for AWS CodeCommit and click Generate credentials.



Download and save the credentials.

## HTTPS Git credentials for AWS CodeCommit

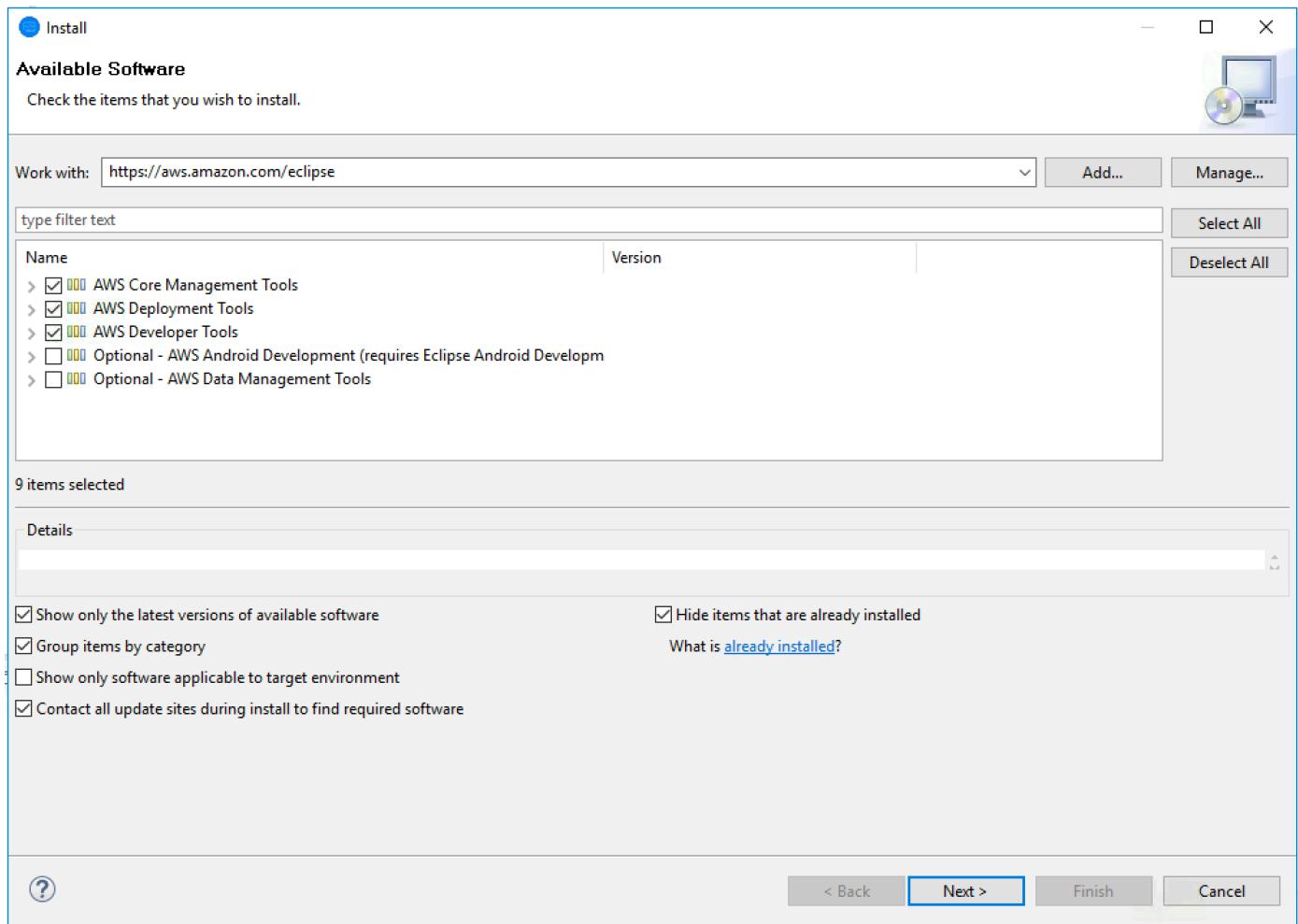
Generate a user name and password you can use to authenticate HTTPS connections to AWS CodeCommit repositories. You can generate and store up to 2 sets of credentials. [Learn more](#)

	User name	Status	Created
<input type="radio"/>	CICD-User-at-954254376221	Active	2020-02-10 18:39 EST

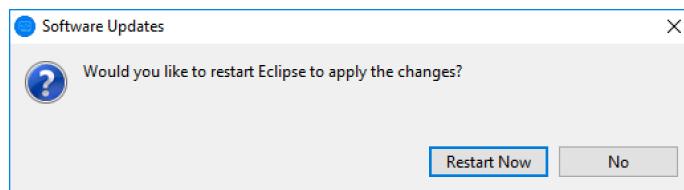
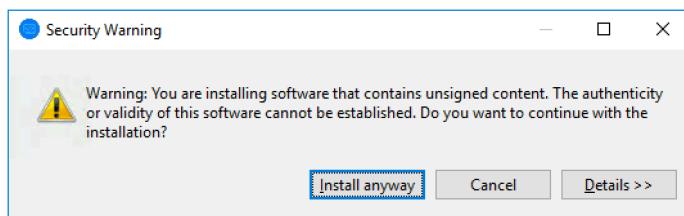
Back to the MF EDz instance, Install the AWS Toolkit for Eclipse with the following instructions.  
Instructions can be found here: <https://aws.amazon.com/eclipse/>

Within Eclipse, click Help and then click Install New Software.

In the Work with field, type `https://aws.amazon.com/eclipse` and then press Enter.

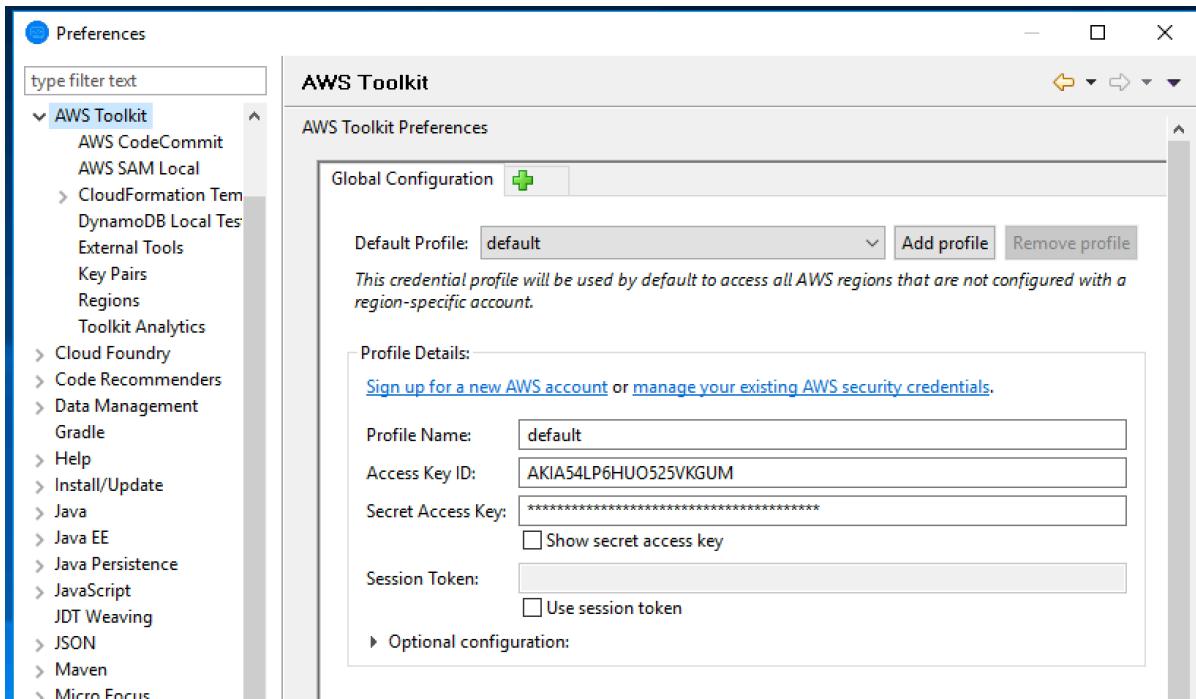


Select the main AWS components and click Next, then review packages, accept license and install.

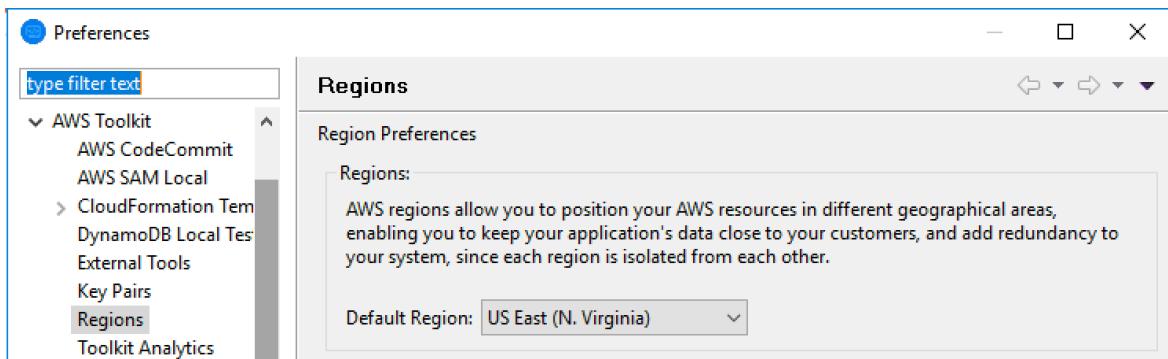


After restart, enter the Access Key and Secret Access Key for the CICD-User.

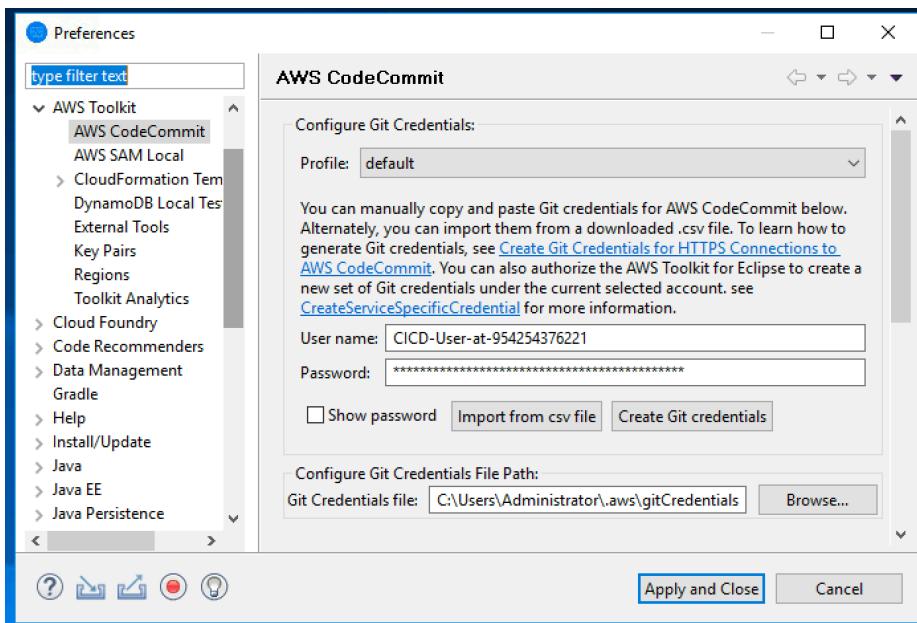
It will then show in the Eclipse Preferences



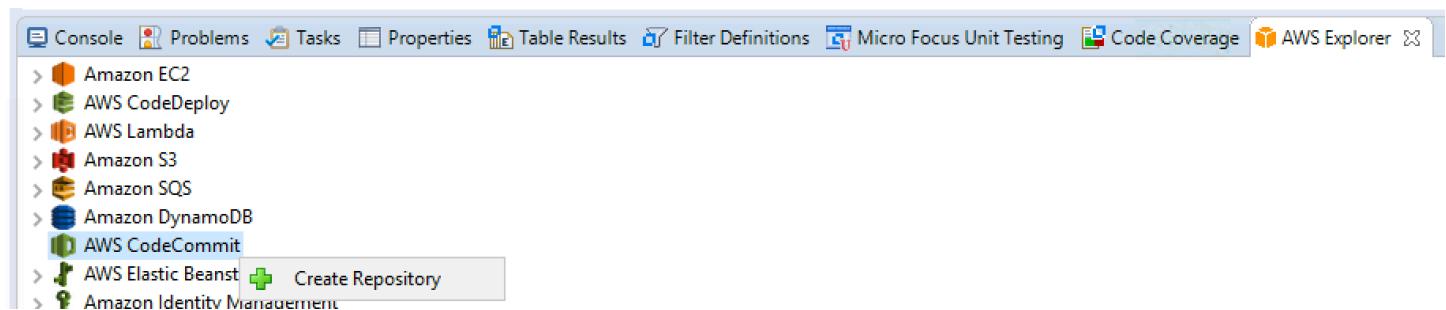
Under AWS Toolkit => Regions, verify or set your desired region for your setup.



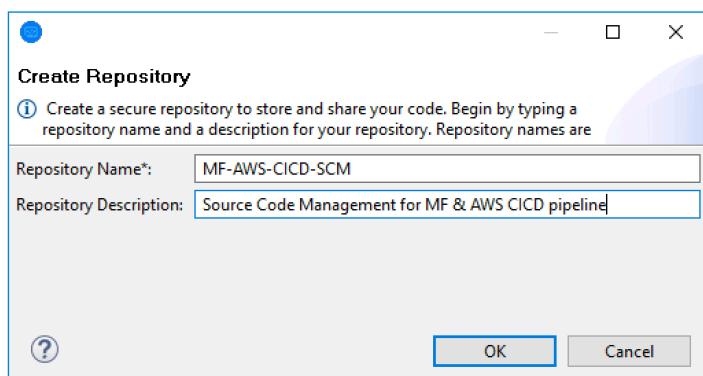
Expand the AWS Toolkit menu and choose AWS CodeCommit. Enter the user name and password for your Git credentials importing them from the .csv file. Choose Apply, and then choose OK.



Save with Apply and Close.

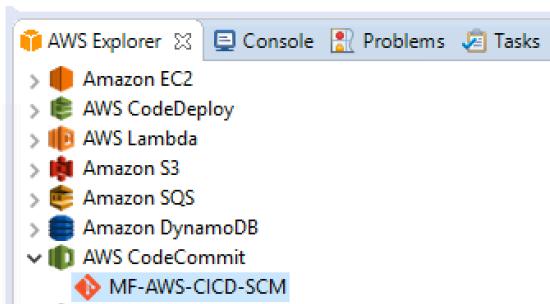


In AWS Explorer, right-click on AWS CodeCommit and click Create Repository.



Enter the Repository Name: MF-AWS-CICD-SCM

Enter the Repository Description: Source Code Management for MF & AWS CICD pipeline



It can be opened in CodeCommit Repository Editor.

This screenshot shows the AWS CodeCommit Repository Editor for the 'MF-AWS-CICD-SCM' repository. The top bar has a tab for 'MF-AWS-CICD-SCM'. Below it, there are fields for Last Modified Date (Tue Feb 11 01:49:09 UTC 2020), Repository Description (Source Code Management for MF & AWS CICD pipeline), Clone URL HTTPS (https://git-codecommit.us-east-1.amazonaws.com/v1/repos/MF-AWS-CICD-SCM), and Clone URL SSH (ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/MF-AWS-CICD-SCM). A 'Check out' button is present. Below these fields is a section titled 'Commit History for Branch:' with a dropdown menu. The main area displays a table with columns for Commit ID, Message, and Committer, though no actual commits are listed.

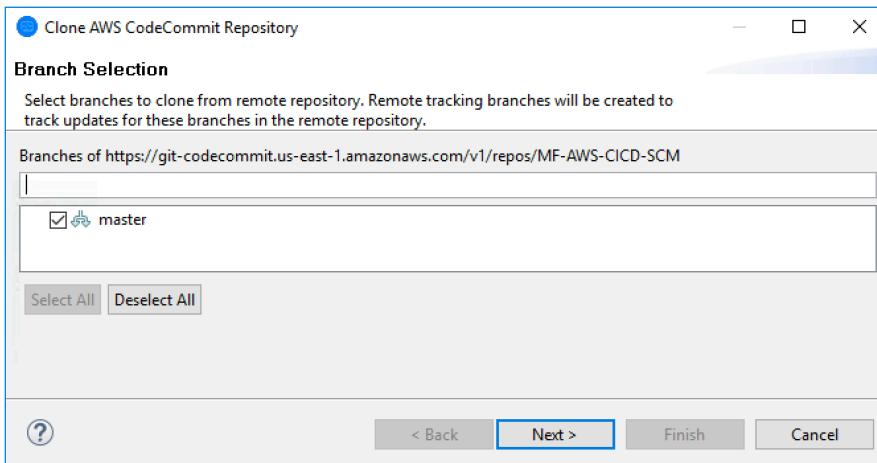
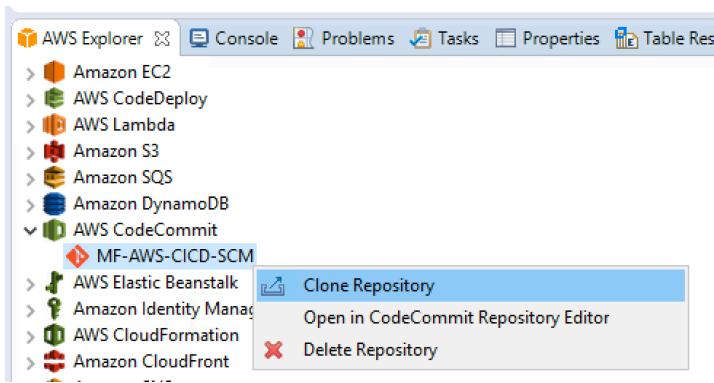
And also shows in AWS console:

This screenshot shows the AWS CodeCommit AWS console. The top navigation bar includes 'Services', 'Resource Groups', and 'Admin/phvalenc-Isengard @ 9... N. Virginia Support'. The left sidebar under 'Developer Tools' shows 'CodeCommit' selected. In the main content area, the 'Repositories' section is displayed. A table lists one repository: 'MF-AWS-CICD-SCM' with a description of 'Source Code Management for MF & AWS CICD pipeline', last modified 2 minutes ago, and clone URLs for HTTPS and SSH.

In AWS CodeCommit, create a dummy file (this creates the master branch in the repository).

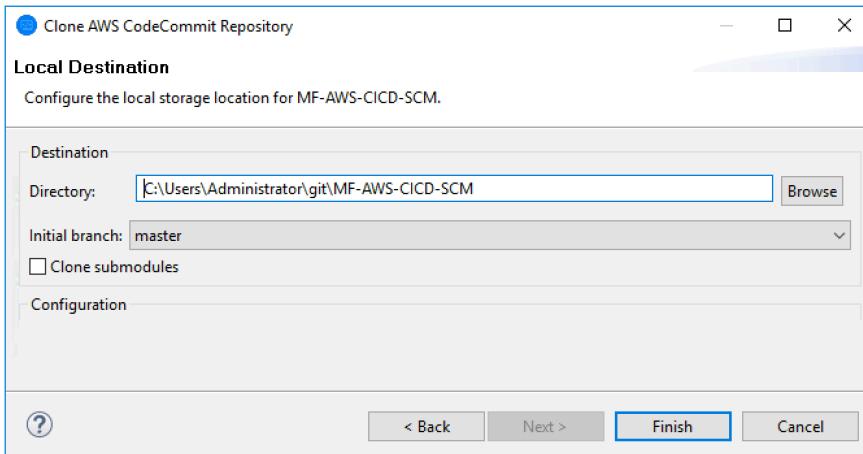
This screenshot shows the AWS CodeCommit AWS console. The left sidebar under 'Developer Tools' shows 'CodeCommit' selected. In the main content area, the 'MF-AWS-CICD-SCM' repository is selected. On the right, a modal window titled 'MF-AWS-CICD-SCM' shows the 'Code' section. It displays a table with a single row for a file named 'dummy'.

In Eclipse AWS Explorer choose Clone Repository

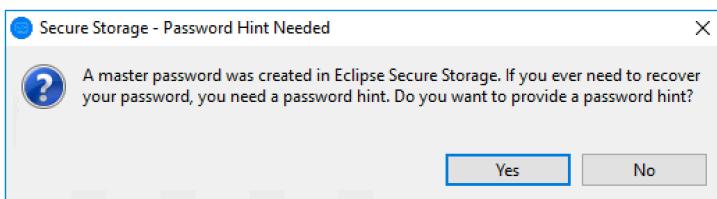


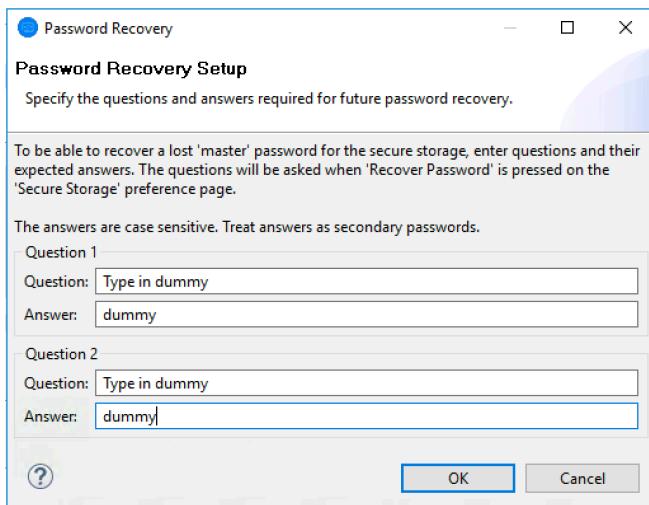
The master branch was created by the dummy file.

Click Next.



Click Finish.



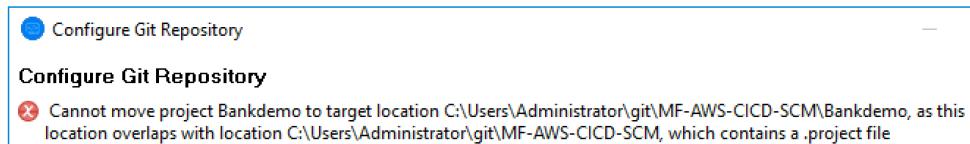


## 3.2 SOURCE – MF EDz to AWS CodeCommit code push

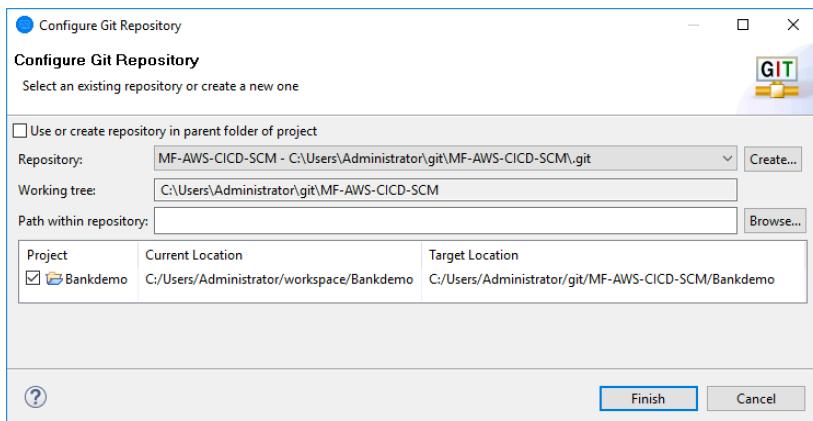
Now that we have a local clone of our repository, we're ready to start putting the BankDemo source code into the Git local clone repository.

Select BankDemo project and use Team -> Share Project... to connect that project with the repository we just cloned.

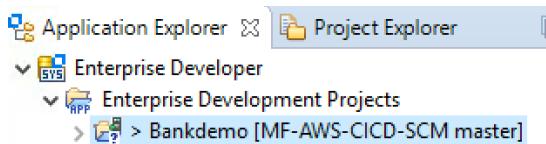
Select the local Git repository in the working tree: C:\Users\Administrator\git\MF-AWS-CICD-SCM



If you see the above error message, delete the C:\Users\Administrator\git\MF-AWS-CICD-SCM\.project file and redo Select BankDemo project and use Team -> Share Project...



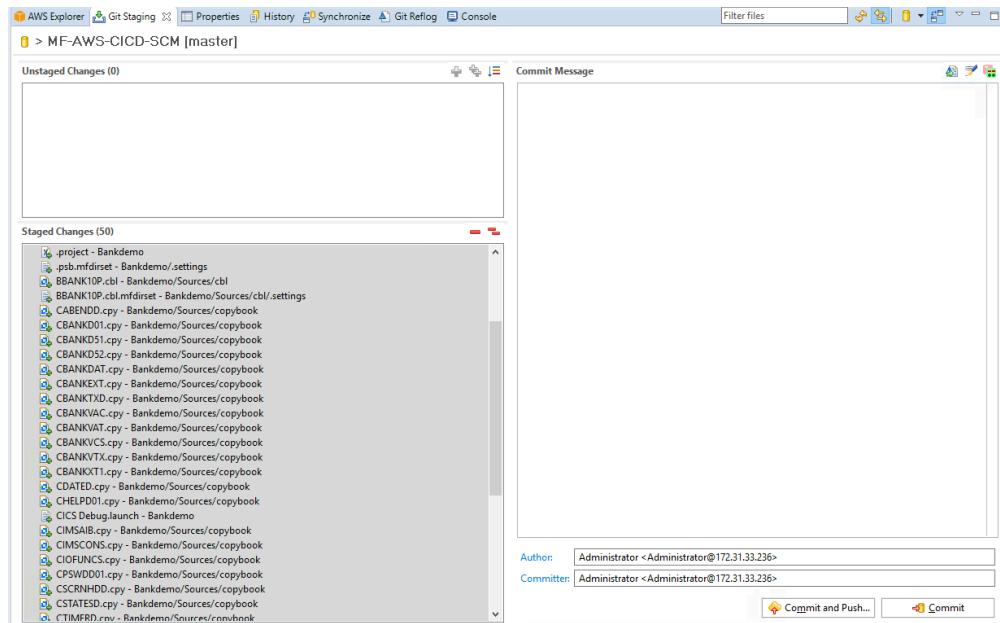
Click Finish. Bankdemo will then recompile/rebuild. You will then notice the Bankdemo project associated with the cloned local Git.



On the Bankdemo project, select Team -> Commit



Add all Unstaged Changes to Staged Changes

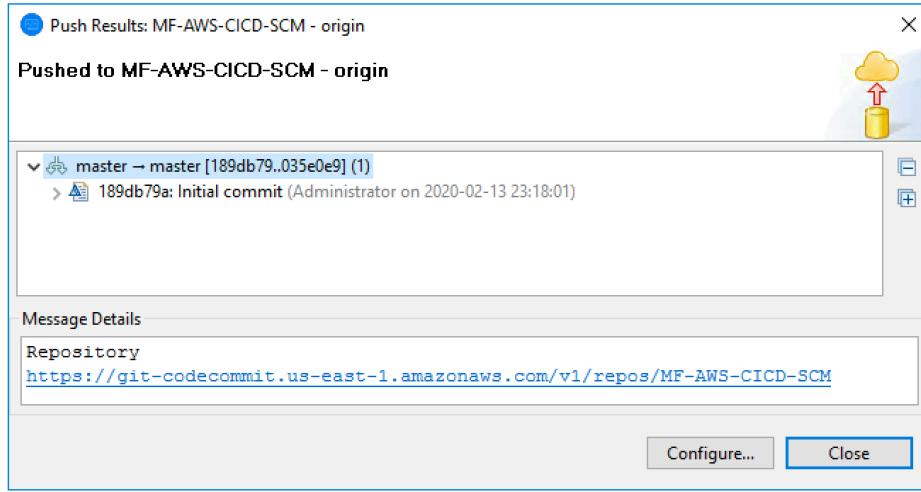


Enter a Commit Message and click Commit.

Changes are committed to the local cloned Git repository.

On the Bankdemo project, select Team -> Push to Upstream to push the source code to CodeCommit.





You can then see the code pushed to CodeCommit via the Repository Editor

The screenshot shows the AWS CodeCommit Repository Editor for the repository MF-AWS-CICD-SCM. It displays basic repository details: Last Modified Date (Thu Feb 13 23:19:19 UTC 2020), Repository Description (Source Code Management for MF & AWS CICD pipeline), Clone URL Https (<https://git-codecommit.us-east-1.amazonaws.com/v1/repos/MF-AWS-CICD-SCM>), and Clone URL SSH (`ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/MF-AWS-CICD-SCM`). A "Check out" button is visible. Below, a "Commit History for Branch: master" table shows one commit:

Commit ID	Message	Committer	Date
<code>189db79a1610809ef07ef6010e285cd2066932d0 035e0e9a10428fb4348280d1d0c20ccaf597001d</code>	Initial commit Added dummy	Administrator dummy	02/13/2020 23:18:01 02/11/2020 20:06:22

And you can see all the files from AWS Console in CodeCommit:

The screenshot shows the AWS Management Console interface for CodeCommit. The left sidebar shows "Developer Tools" and "CodeCommit". Under "Source", "CodeCommit" is selected. The main area shows the repository "MF-AWS-CICD-SCM". The "Sources" tab is selected, displaying a list of files and folders: "..", "bms", "cbl", "copybook", and "jcl". The navigation bar at the top includes "Developer Tools", "CodeCommit", "Repositories", and "MF-AWS-CICD-SCM".

# 4 BUILD stage

---

## 4.1 BUILD – Amazon ECR docker image with Micro Focus build tools

For getting access to Micro Focus software, please contact your Micro Focus representative or contact Micro Focus following this link: <https://www.microfocus.com/en-us/contact/contactme>

For this section we need the Micro Focus Build tools packaged as Docker containers. In our example, these containers are initially stored in AWS S3.

Create an EC2-to-S3-ECR-microfocus-aws IAM role with inline policy

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:Get*",  
                "s3>List*"  
            ],  
            "Resource": "arn:aws:s3:::microfocus-aws/*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "ecr>ListTagsForResource",  
                "ecr:UploadLayerPart",  
                "ecr>ListImages",  
                "ecr:CompleteLayerUpload",  
                "ecr:DescribeRepositories",  
                "ecr:BatchCheckLayerAvailability",  
                "ecr:GetLifecyclePolicy",  
                "ecr:DescribeImageScanFindings",  
                "s3>ListAccessPoints",  
                "ecr:GetLifecyclePolicyPreview",  
                "ecr:GetDownloadUrlForLayer",  
                "ecr:GetAuthorizationToken",  
                "ecr:PutImage",  
                "ecr:BatchGetImage",  
                "ecr:DescribeImages",  
                "ecr:InitiateLayerUpload"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

Launch a new EC2 instance with “Microsoft Windows Server 2016 Base with Containers” (because AWS CodeBuild executes Windows Server containers using Windows Server 2016 hosts). This instance is temporary to prepare and push the container image to Amazon ECR.



## Include the EC2-to-S3-ECR-microfocus-aws role

IAM role EC2-to-S3-ECR-microfocus-aws

## Allocate 256 GB of disk space

### Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)
Root	/dev/sda1	snap-02b94dd338b45921e	256	General Purpose SSD (gp2)	100 / 3000	N/A

Log on to the instance.

Start Command Prompt as an administrator.

Check the docker version in your instance:

```
docker version
Client:
  Version: 1.12.2-cs2-ws-beta
  API version: 1.25
  Go version: go1.7.1
  Git commit: 050b611
  Built: Tue Oct 11 02:35:40 2016
  OS/Arch: windows/amd64

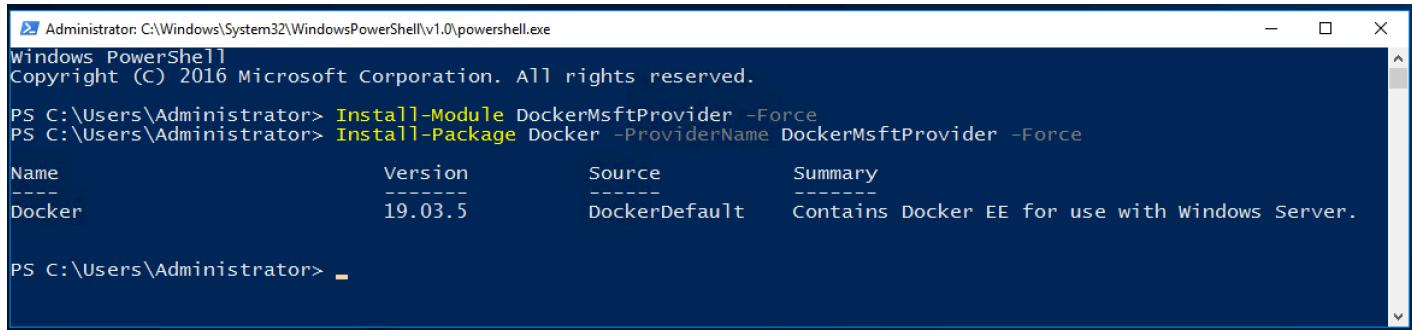
Server:
  Version: 1.12.2-cs2-ws-beta
  API version: 1.25
  Go version: go1.7.1
  Git commit: 050b611
  Built: Tue Oct 11 02:35:40 2016
  OS/Arch: windows/amd64
```

If the docker version is not Docker Enterprise v19 (like the above is not the proper version), we need to install it with the following instructions:

```
start powershell
```

In the new Administrator PowerShell windows, enter commands:

```
Install-Module DockerMsftProvider -Force
Install-Package Docker -ProviderName DockerMsftProvider -Force
```



The screenshot shows an Administrator PowerShell window with the title bar "Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe". The command "Install-Module DockerMsftProvider -Force" is run, followed by "Install-Package Docker -ProviderName DockerMsftProvider -Force". The output shows the Docker module installed with version 19.03.5 from the DockerDefault provider. The summary indicates it contains Docker EE for use with Windows Server.

```
Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> Install-Module DockerMsftProvider -Force
PS C:\Users\Administrator> Install-Package Docker -ProviderName DockerMsftProvider -Force

Name          Version      Source       Summary
----          -----      -----       -----
Docker        19.03.5    DockerDefault Contains Docker EE for use with Windows Server.

PS C:\Users\Administrator>
```

Start Docker Enterprise

```
net start docker
The Docker Engine service is starting.
The Docker Engine service was started successfully.
```

Check the new docker version in your instance:

```
docker version
Client: Docker Engine - Enterprise
  Version:           19.03.5
  API version:      1.40
  Go version:       go1.12.12
  Git commit:       2ee0c57608
  Built:            11/13/2019 08:00:16
  OS/Arch:          windows/amd64
  Experimental:    false

Server: Docker Engine - Enterprise
  Engine:
    Version:          19.03.5
    API version:     1.40 (minimum version 1.24)
    Go version:      go1.12.12
    Git commit:      2ee0c57608
    Built:           11/13/2019 07:58:51
    OS/Arch:         windows/amd64
    Experimental:   false
```

Docker Enterprise v19 is good and we can proceed.

Reboot the instance.

Download AWS CLI installer from <https://s3.amazonaws.com/aws-cli/AWSCLI64PY3.msi>

Run CLI Setup installer.

Then copy the Micro Focus docker container file from your S3 bucket:

```
aws s3 cp s3://microfocus-aws/ed_build_tools_dockerfiles_5.0_windows_pu05.zip .
download: s3://microfocus-aws/ed_build_tools_dockerfiles_5.0_windows_pu05.zip to
./ed_build_tools_dockerfiles_5.0_windows_pu05.zip
```

Extract the zip file.

In the extracted location, copy in the correct license file (.mflic) for that product

Administrator > ed_build_tools_dockerfiles_5.0_windows_pu05				
	Name	Date modified	Type	Size
ess	Dockerfile.x86	1/10/2019 2:21 PM	X86 File	1 KB
nts	edbt_for_docker_50_pu05_250319	1/14/2020 8:22 PM	Application	463,518 KB
ds	EDBuildToolsDocker.mflic	2/13/2020 3:30 PM	MFLIC File	5 KB
	getAdoptOpenJDK	11/12/2018 10:56 ...	Windows PowerS...	5 KB
	palic.env	11/8/2017 4:04 PM	ENV File	1 KB
	prodver.env	1/9/2020 5:51 PM	ENV File	1 KB
	README	4/8/2019 8:57 AM	HTML Document	12 KB

Also copy the C:\MicroFocus\_Software\EDBT\edbt\_50.exe from the EDz instance to the extracted folder.

From the extracted folder, run bld.bat IacceptEULA which will build the container images for you

```
bld.bat IacceptEULA
```

If you receive an error message such as the following, you likely don't have Docker Enterprise v19 installed and see previous steps to install it.

```
Step 1/26 : ARG BASE_SUFFIX=
Please provide a source image with `from` prior to commit
```

```

Administrator: Command Prompt
C:\EDTools>SETX /m TXDIR "C:\EDTools\\"

SUCCESS: Specified value was saved.
Removing intermediate container 51cde586775f
--> 7c78016e6fae
Step 6/7 : RUN setx /M PATH "%PATH%;C:\\Windows\\Microsoft.NET\\Framework64\\v4.0.30319"
--> Running in df463355ead1

SUCCESS: Specified value was saved.
Removing intermediate container df463355ead1
--> c693e9440bee
Step 7/7 : LABEL com.microfocus.third_parties.dotnet=4.7.2
--> Running in 1be6d434ab0b
Removing intermediate container 1be6d434ab0b
--> 948ea87022be
Successfully built 948ea87022be
Successfully tagged microfocus/edbbuildtools-build:win_5.0_x64

Complete - We have the following microfocus/edbbuildtools images
microfocus/edbbuildtools-build    win_5.0_x64          948ea87022be      1 second ago     18.9GB
microfocus/edbbuildtools-build    win_5.0_x86          7b3542d7d007      25 seconds ago   18.9GB
microfocus/edbbuildtools-build    win_5.0             bfd78b3a744a      58 seconds ago   18.8GB
microfocus/edbbuildtools         win_5.0_x64          fe1cb18df2da      18 minutes ago   16.3GB
microfocus/edbbuildtools         win_5.0_x86          436dfcb8a69b      18 minutes ago   16.3GB
microfocus/edbbuildtools         win_5.0             512178790dfa      50 minutes ago   16.2GB

C:\Users\Administrator\ed_build_tools_dockerfiles_5.0_windows_pu05>

```

## Build output:

```

Complete - We have the following microfocus/edbbuildtools images
microfocus/edbbuildtools-build    win_5.0_x64          948ea87022be      1 second ago     18.9GB
microfocus/edbbuildtools-build    win_5.0_x86          7b3542d7d007      25 seconds ago   18.9GB
microfocus/edbbuildtools-build    win_5.0             bfd78b3a744a      58 seconds ago   18.8GB
microfocus/edbbuildtools         win_5.0_x64          fe1cb18df2da      18 minutes ago   16.3GB
microfocus/edbbuildtools         win_5.0_x86          436dfcb8a69b      18 minutes ago   16.3GB
microfocus/edbbuildtools         win_5.0             512178790dfa      50 minutes ago   16.2GB

```

You can then display images with:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
microfocus/edbbuildtools-build	win_5.0_x64	948ea87022be	25 minutes ago	18.9GB
microfocus/edbbuildtools-build	win_5.0_x86	7b3542d7d007	25 minutes ago	18.9GB
microfocus/edbbuildtools-build	win_5.0	bfd78b3a744a	25 minutes ago	18.8GB
microfocus/edbbuildtools	win_5.0_x64	fe1cb18df2da	43 minutes ago	16.3GB
microfocus/edbbuildtools	win_5.0_x86	436dfcb8a69b	43 minutes ago	16.3GB
microfocus/edbbuildtools	win_5.0	512178790dfa	About an hour ago	16.2GB
microsoft/dotnet-framework	4.7.2-runtime	da6e6a287bce	4 weeks ago	13.3GB
microsoft/dotnet-framework	4.7.2-sdk	bab65b1f870b	4 months ago	15.8GB
hello-world	nanoserver-sac2016	2c911f8d79db	13 months ago	1.17GB

Explore the edbuildtools-build win\_5.0\_x64 docker image with commands similar to the followings:

```

docker run -it 948ea87022be cmd
dir
exit

```

In Amazon ECR, create a repository to store the docker image.

**Repository configuration**

Repository name  
954254376221.dkr.ecr.us-east-1.amazonaws.com/ **mf-aws-cicd-container-repository**  
A namespace can be included with your repository name (e.g. namespace/repo-name).

Tag immutability  
Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.  
 Disabled

Scan on push  
Enable scan on push to have each image automatically scanned after being pushed to a repository. If disabled, each image scan must be manually started to get scan results.  
 Disabled

**Create repository**

Enter repository name: **mf-aws-cicd-container-repository**, then select **Create repository**.

Successfully created repository **mf-aws-cicd-container-repository**

Repository name	URI	Created at	Tag immutability	Scan on push
<a href="#">mf-aws-cicd-container-repository</a>	954254376221.dkr.ecr.us-east-1.amazonaws.com/mf-aws-cicd-container-repository	02/13/20, 09:17:45 PM	Disabled	Disabled

**Create repository**

Click **View push commands**

## Push commands for mf-aws-cicd-container-repository

X

macOS / Linux

Windows

Ensure you have installed the latest version of the AWS CLI and Docker. For more information, see the [ECR documentation](#).

1. Retrieve the login command to use to authenticate your Docker client to your registry.

Use AWS Tools for PowerShell:

```
Invoke-Expression -Command (Get-ECRLoginCommand -Region us-east-1).Command
```

2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions [here](#). You can skip this step if your image is already built:

```
docker build -t mf-aws-cicd-container-repository .
```

3. After the build completes, tag your image so you can push the image to this repository:

```
docker tag mf-aws-cicd-container-repository:latest 954254376221.dkr.ecr.us-east-1.amazonaws.com/mf-aws-cicd-container-repository:latest
```

4. Run the following command to push this image to your newly created AWS repository:

```
docker push 954254376221.dkr.ecr.us-east-1.amazonaws.com/mf-aws-cicd-container-repository:latest
```

Close

Copy-paste the commands which will be customized to push the docker image from the temporary EC2 instance to Amazon ECR.

```
Invoke-Expression -Command (Get-ECRLoginCommand -Region us-east-1).Command
docker build -t mf-aws-cicd-container-repository .
docker tag mf-aws-cicd-container-repository:latest 954254376221.dkr.ecr.us-east-1.amazonaws.com/mf-aws-cicd-container-repository:latest
docker push 954254376221.dkr.ecr.us-east-1.amazonaws.com/mf-aws-cicd-container-repository:latest
```

Click on the mf-aws-cicd-container-repository name, then select Permissions.

Amazon ECR

Repositories

Images

Permissions

Lifecycle Policy

Tags

Under Permissions – Statements, click Edit to create one, then choose Add statement.

Enter CodeBuildAccess for the Statement name.

Enter codebuild.amazonaws.com for the Service principal.

**CodeBuildAccess**

**Delete**

**Statement name**  
CodeBuildAccess

**Effect**  
Specifies whether the statement results in an allow or an explicit deny.  
 Allow  
 Deny

**Principal**  
The entities (AWS service, IAM user, role, group, AWS account ID, or Everyone) you want the statement to apply to. For more information, see [Principal](#).  
 Everyone (\*)

**Service principal - optional**  
The service principal to apply the statement to.  
codebuild.amazonaws.com

Comma delimited list

**AWS account IDs - optional**  
The AWS account(s) to apply the statement to. All users under the AWS account will be affected.

Comma delimited list

For Actions, select the pull-only actions ecr:GetDownloadUrlForLayer, ecr:BatchGetImage, and ecr:BatchCheckLayerAvailability.

**Actions**  
The API actions to apply to the statement.

Add another option ▾

ecr:BatchCheckLayerAvailability X    ecr:BatchGetImage X    ecr:GetDownloadUrlForLayer X

**Cancel** **Save**

Click Save.

ECR > Repositories > mif-aws-cicd-container-repository > Permissions

**Permissions**

**Edit policy JSON** **Edit**

**CodeBuildAccess**

<b>Effect</b> Allow	<b>Principal</b> -
<b>Service principals</b> codebuild.amazonaws.com	<b>AWS Account IDs</b> -
<b>Actions</b>	
ecr:BatchCheckLayerAvailability ecr:BatchGetImage ecr:GetDownloadUrlForLayer	

Back in the temporary EC2 instance with the build images, start a PowerShell window and run this command to login to Amazon ECR:

```
Invoke-Expression -Command (Get-ECRLoginCommand -Region us-east-1).Command
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in C:\Users\Administrator\.docker\config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
```

From the list of images previously displayed, find the image ID for the microfocus/edbbuildtools-build-win\_5.0\_x64 image which is 948ea87022be in our example). Then incorporate the image ID in the following command and run the command to tag the image:

```
docker tag 948ea87022be 954254376221.dkr.ecr.us-east-1.amazonaws.com/mf-aws-cicd-container-
repository:edbbuildtools-build-win_5.0_x64
```

Then push this image in the Amazon ECR repository:

```
docker push 954254376221.dkr.ecr.us-east-1.amazonaws.com/mf-aws-cicd-container-
repository:edbbuildtools-build-win_5.0_x64
```

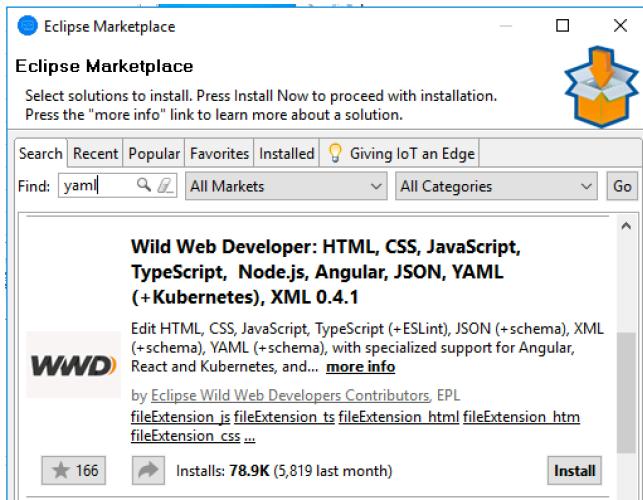
```
Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Error response from daemon: No such image: 948ea87022be:latest
PS C:\Users\Administrator\ed_build_tools_dockerfiles_5.0_windows_pu05> docker tag 948ea87022be 954254376221.dkr.ecr.us-e
ast-1.amazonaws.com/mf-aws-cicd-container-repository:latest
PS C:\Users\Administrator\ed_build_tools_dockerfiles_5.0_windows_pu05> docker push 954254376221.dkr.ecr.us-east-1.amazon
aws.com/mf-aws-cicd-container-repository:latest
The push refers to repository [954254376221.dkr.ecr.us-east-1.amazonaws.com/mf-aws-cicd-container-repository]
3bb63ff73974: Pushed
4cf7f1f1609e: Pushed
03d0b166ee1f: Pushed
924f9114294a: Pushed
abb434474ce5: Pushed
e8bdc1539e8: Pushed
5704972ecd37: Pushed
60bdf08061c5: Pushed
c3a264b1c6a0: Pushed
449bdac3791a: Pushed
45bdc078ee64: Pushed
669c117306fc: Pushed
4e04013455ed: Pushed
b8607e6e8c90: Pushed
edab4572a798: Pushed
75d493098914: Pushed
684daf9a5db3: Pushed
b3854b38232c: Pushed
09e5e664105f: Pushed
a268f0f3a56d: Pushed
d1a4ad9c6f8b: Pushed
1f1c85ef5f07: Pushed
7cc883a81f09: Pushed
9f1349de7d76: Pushed
b7615f227358: Pushed
e3c2a294912b: Pushed
6440b624863b: Pushed
5e85a2669826: Pushed
de3e408934b7: Pushed
f56cc0b09e46: Pushed
18f6a03f3100: Pushed
b79439598ebc: Pushed
cc549fe6f383: Pushed
d827ca1d798: Pushed
d9d89d66e8b3: Pushed
a3e428609ea3: Pushed
417d98384a3: Pushed
914c106d1b66: Pushed
7cd944f2342c: Pushed
e0463cfab0b4: Pushed
76cdc23ced7a: Skipped foreign layer
f358be10862c: Skipped foreign layer
latest: digest: sha256:6d54b52e44660212dc32a49d150436ec81e6c5dbde65e2a36305fe7b786f13d8 size: 9345
PS C:\Users\Administrator\ed_build_tools_dockerfiles_5.0_windows_pu05>
```

On the Amazon ECR side, you can then see the new image in the repository

Image tag	Image URI	Pushed at	Digest	Size (MB)	Scan status	Vulnerabilities
edbbuildtools-build-win_5.0_x64	954254376221.dkr.ecr.us-east-1.amazonaws.com/mf-aws-cicd-container-repository:edbbuildtools-build-win_5.0_x64	02/14/20, 06:05:10 PM	sha256:6d54b52e4...	3562.22	-	-

## 4.2 BUILD – AWS CodeBuild configuration

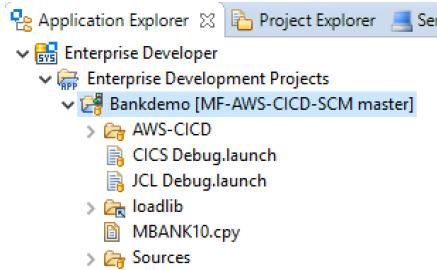
In EDz, select Help -> Eclipse Marketplace... Enter yaml in Find field and search.



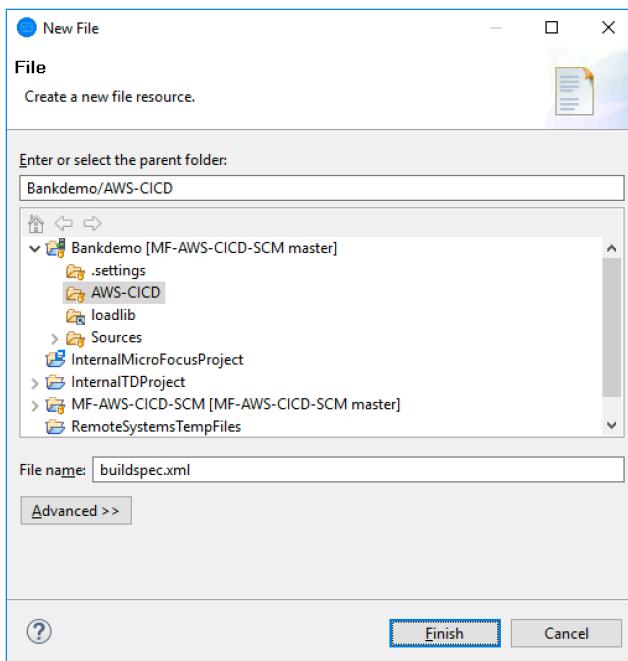
Install Wild Web Developer. It facilitates the editing of Yaml files that we will create next.

In EDz Bankdemo project, select New -> Folder -> Folder... and create an AWS-CICD folder.

You may need to Refresh Resource (F5) to see it.



In this AWS-CICD folder, select New -> File -> Other File... and create a buildspec.yml file.



The buildspec.yml file is a build specification file which contains a collection of build commands and related settings, in YAML format, that CodeBuild uses to run a build.

Details about the buildspec.yml file and syntax are here:

<https://docs.aws.amazon.com/codebuild/latest/userguide/build-spec-ref.html>

Buildspec.yml commands run in a Windows Server Core 2016 image using the Powershell shell.

Edit the buildspec.yml file:

```

version: 0.2

env:
  variables:
    SOURCE_COBOL_FOLDER: .\Bankdemo\Sources\cbl
    AWS_CICD_FOLDER: .\Bankdemo\AWS-CICD
    COBCPY: .\Bankdemo\Sources\copybook;C:\EDTools\cpylib # where to find copybooks such as DFHAID and
    application copybooks
    COBOL_EXE_PATH_FILE: C:\EDTools\bin\cobol.exe
    CBLLINK_EXE_PATH_FILE: C:\EDTools\bin\cbllink.exe
    COBOL_COMPILER_DIRECTIVES:
      "preprocess(EXCI),CICSECM(),CHARSET(EBCDIC),DIALECT(ENTCOBOL),SOURCEFORMAT(fixed),NOPANVALET,NOLIBRARIAN
      ,ANIM,EXITPROGRAM(ANSI),WARNING(1),MAX-ERROR(100)"
      COBDIR: C:\EDTools # where to find the message file

phases:
  pre_build:
    commands:
      - echo =====
      - echo Variables
      - 'echo SOURCE_COBOL_FOLDER: $env:SOURCE_COBOL_FOLDER'
      - 'echo AWS_CICD_FOLDER: $env:AWS_CICD_FOLDER'
      - 'echo COBOL_COMPILER_DIRECTIVES: $env:COBOL_COMPILER_DIRECTIVES'
      - 'echo COBOL_EXE_PATH_FILE: $env:COBOL_EXE_PATH_FILE'
      - 'echo CBLLINK_EXE_PATH_FILE: $env:CBLLINK_EXE_PATH_FILE'
      - 'echo COBDIR: $env:COBDIR'
      - 'echo COBCPY: $env:COBCPY'
      - dir
      - dir $env:SOURCE_COBOL_FOLDER

  build:
    commands:
      - echo =====
      - echo "Compiling begins"
      - foreach ($f in Get-ChildItem $env:SOURCE_COBOL_FOLDER -Filter "*.cbl") {

```

```

echo -----
echo "Compiling $($f.FullName)";
& $env:COBOL_EXE_PATH_FILE "$($f.FullName),,$($env:COBOL_COMPILER_DIRECTIVES);";
}
-----
- echo -----
- echo "Compiling ends"
- echo ""
- echo =====
- echo "Linking begins"
- foreach ($f in Get-ChildItem "." -Filter "*.obj") {
    echo -----
    echo "Linking $($f.FullName)";
    & $env:CBLLINK_EXE_PATH_FILE -d $f.FullName;
}
-----
- echo -----
- echo "Linking ends"

post_build:
  commands:
    - echo =====
    - echo "Preparing build output files for packaging"
    - mkdir ./dll
    - cp *.dll ./dll
    - cp $env:AWS_CICD_FOLDER/* .
    - compress-archive -path ./dll -destinationpath ./deploy.zip
    - compress-archive -path $env:AWS_CICD_FOLDER/* -update -destinationpath ./deploy.zip

artifacts:
  files:
    - deploy.zip          # For CodeDeploy manual deployment
    - appspec.yml         # For CodePipeline automated deployment
    - codedeploy-before-install.bat # For CodePipeline automated deployment
    - codedeploy-after-install.bat # For CodePipeline automated deployment
    - dll/*               # For CodePipeline automated deployment
  discard-paths: no

```

Save this buildspec.yml file.

Commit and push this change to CodeCommit.

In S3, create a S3 bucket for the build artefact. We use mf-aws-cicd-artifacts in our example.

<input type="checkbox"/>	 mf-aws-cicd-artifacts	Bucket and objects not public	US East (N. Virginia)
--------------------------	---	----------------------------------	--------------------------

Go to CodeBuild

Click Create build project.

## Create build project

**Project configuration**

Project name  
MF-AWS-CICD-Build  
A project name must be 2 to 255 characters. It can include the letters A-Z and a-z, the numbers 0-9, and the special characters - and \_.

Description - *optional*  
  
Build badge - *optional*  
 Enable build badge

▶ Additional configuration  
tags

Enter the project name: MF-AWS-CICD-Build.

**Source**

Add source

Source 1 - Primary

Source provider  
AWS CodeCommit

Repository  
MF-AWS-CICD-SCM

Reference type  
Choose the source version reference type that contains your source code.  
 Branch  
 Git tag  
 Commit ID

Branch  
Choose a branch that contains the code to build.  
master

Commit ID - *optional*  
Choose a commit ID. This can shorten the duration of your build.  
refs/heads/master

Source version info  
745795a8 New buildspec.yml

▶ Additional configuration  
Git clone depth, Git submodules

Select CodeCommit provider, MF-AWS-CICD-SCM repository, master branch.

**Environment**

Environment image

Managed image  
Use an image managed by AWS CodeBuild

Custom image  
Specify a Docker image

Environment type  
Choose an environment type

Windows

Image registry

Amazon ECR  
Use an image from Amazon ECR

Other registry  
Use an image hosted in an external Docker registry

ECR account  
You can use an ECR image from your account or another that you have access to.

My ECR account

Other ECR account

Amazon ECR repository  
Choose an Amazon ECR repository

mf-aws-cicd-container-repository

Amazon ECR image  
Choose an Amazon ECR image

edbbuildtools-build-win\_5.0\_x64

Image pull credentials  
Choose which service role will be authorized to pull the selected image

AWS CodeBuild credentials  
Use the AWS CodeBuild default service role

Project service role  
Use the service role associated with this project to pull the image

Privileged

Enable this flag if you want to build Docker images or want your builds to get elevated privileges

Service role

New service role  
Create a service role in your account

Existing service role  
Choose an existing service role from your account

Role name

codebuild-mf-aws-cicd-build-service-role

Type your service role name

► Additional configuration  
Timeout, certificate, VPC, compute type, environment variables, file systems

Select a Custom image (Docker image) of Windows type within the Amazon ECR registry in My ECR account within mf-aws-cicd-container-repository named edbbuildtools-build-win\_5.0\_x64. We use AWS CodeBuild credentials and a New service role named codebuild-mf-aws-cicd-build-service-role.

**Buildspec**

Build specifications

Use a buildspec file  
Store build commands in a YAML-formatted buildspec file

Insert build commands  
Store build commands as build project configuration

Buildspec name - optional  
By default, CodeBuild looks for a file named buildspec.yml in the source code root directory. If your buildspec file uses a different name or location, enter its path from the source root here (for example, buildspec-two.yml or configuration/buildspec.yml).

Bankdemo/AWS-CICD/buildspec.yml

We use a buildspec file in the following location: Bankdemo/AWS-CICD/buildspec.yml

Under Artifacts, choose the Amazon S3 and mf-aws-cicd-artifacts bucket.

**Artifacts**

Add artifact

Artifact 1 - Primary

Type

Amazon S3

You might choose no artifacts if you are running tests or pushing a Docker image to Amazon ECR.

Bucket name

mf-aws-cicd-artifacts

Name

The name of the folder or compressed file in the bucket that will contain your output artifacts. Use Artifacts packaging under Additional configuration to choose whether to use a folder or compressed file. If the name is not provided, defaults to project name.

For the Logs, we store build output logs in CloudWatch.

**Logs**

CloudWatch

CloudWatch logs - *optional*  
Checking this option will upload build output logs to CloudWatch.

Group name

Stream name

Click Create build project.

Developer Tools > CodeBuild > Build projects > MF-AWS-CICD-Build

**MF-AWS-CICD-Build**

Notify Share Edit Delete build project Start build

Configuration			
Source provider AWS CodeCommit	Primary repository <a href="#">MF-AWS-CICD-SCM</a>	Artifacts upload location <a href="#">mf-aws-cicd-artifacts</a>	Build badge Disabled

Click Start build.

## MF-AWS-CICD-Build

Notify ▾ Share Edit ▾ Delete build project Start build

### Configuration

Source provider AWS CodeCommit	Primary repository MF-AWS-CICD-SCM	Artifacts upload location mf-aws-cicd-artifacts	Build badge Disabled
-----------------------------------	---------------------------------------	--	-------------------------

**Build history** Build details Build triggers Metrics

#### Build history

C Stop build View artifacts View logs Delete builds Retry build

Build run	Status	Build Number	Source version	Submitter	Duration	Completed
MF-AWS-CICD- Build:79d36a00- 8014-48d5-8348- 898a23874a8c	Succeeded	77	refs/heads/master	Admin/phvalenc- Isengard	1 minute 10 seconds	21 minutes ago

Build logs Phase details Reports Environment variables Build details

Name	Status	Context	Duration	Start time	End time
SUBMITTED	Succeeded	-	<1 sec	Feb 17, 2020 10:20 PM (UTC-5:00)	Feb 17, 2020 10:20 PM (UTC-5:00)
QUEUED	Succeeded	-	<1 sec	Feb 17, 2020 10:20 PM (UTC-5:00)	Feb 17, 2020 10:20 PM (UTC-5:00)
PROVISIONING	Succeeded	-	39 secs	Feb 17, 2020 10:20 PM (UTC-5:00)	Feb 17, 2020 10:21 PM (UTC-5:00)
DOWNLOAD_SOURCE	Succeeded	-	10 secs	Feb 17, 2020 10:21 PM (UTC-5:00)	Feb 17, 2020 10:21 PM (UTC-5:00)
INSTALL	Succeeded	-	<1 sec	Feb 17, 2020 10:21 PM (UTC-5:00)	Feb 17, 2020 10:21 PM (UTC-5:00)
PRE_BUILD	Succeeded	-	5 secs	Feb 17, 2020 10:21 PM (UTC-5:00)	Feb 17, 2020 10:21 PM (UTC-5:00)
BUILD	Succeeded	-	8 secs	Feb 17, 2020 10:21 PM (UTC-5:00)	Feb 17, 2020 10:21 PM (UTC-5:00)
POST_BUILD	Succeeded	-	2 secs	Feb 17, 2020 10:21 PM (UTC-5:00)	Feb 17, 2020 10:21 PM (UTC-5:00)
UPLOAD_ARTIFACTS	Succeeded	-	2 secs	Feb 17, 2020 10:21 PM (UTC-5:00)	Feb 17, 2020 10:21 PM (UTC-5:00)
FINALIZING	Succeeded	-	2 secs	Feb 17, 2020 10:21 PM (UTC-5:00)	Feb 17, 2020 10:21 PM (UTC-5:00)
COMPLETED	Succeeded	-	-	Feb 17, 2020 10:21 PM (UTC-5:00)	-

With a successful build, the artifacts are created in S3.

Amazon S3 > mf-aws-cicd-artifacts > MF-AWS-CICD-Build

### mf-aws-cicd-artifacts

Overview

Type a prefix and press Enter to search. Press ESC to clear.

Upload  + Create folder  Download  Actions ▾

<input type="checkbox"/>	Name ▾
<input type="checkbox"/>	dll
<input type="checkbox"/>	appspec.yml
<input type="checkbox"/>	codedeploy-after-install.bat
<input type="checkbox"/>	codedeploy-before-install.bat
<input type="checkbox"/>	deploy.zip

## 4.3 BUILD – AWS CodePipeline configuration

AWS Services Resource Groups

Developer Tools  X

**CodePipeline**

- ▶ **Source** • CodeCommit
- ▶ **Build** • CodeBuild
- ▶ **Deploy** • CodeDeploy
- ▼ **Pipeline** • CodePipeline
  - Getting started
  - Pipelines**
  - ▶ Settings
- Code pipelines

Developer Tools > CodePipeline > Pipelines

**Pipelines** Info  View pipeline  View history  Delete pipeline  Create pipeline

No results  
There are no results to display.

In CodePipeline, select Pipelines then click Create pipeline

Step 1  
**Choose pipeline settings**

Step 2  
Add source stage

Step 3  
Add build stage

Step 4  
Add deploy stage

Step 5  
Review

Developer Tools > CodePipeline > Pipelines > Create new pipeline

## Choose pipeline settings

### Pipeline settings

#### Pipeline name

Enter the pipeline name. You cannot edit the pipeline name after it is created.

MF-AWS-CICD-Pipeline

No more than 100 characters

#### Service role

New service role

Create a service role in your account

Existing service role

Choose an existing service role from your account

#### Role name

AWSCodePipelineServiceRole-us-east-1-MF-AWS-CICD-Pipeline

Type your service role name

Allow AWS CodePipeline to create a service role so it can be used with this new pipeline

### ▼ Advanced settings

#### Artifact store

Default location

Create a default S3 bucket in your account.

Custom location

Choose an existing S3 location from your account in the same region and account as your pipeline

#### Encryption key

Default AWS Managed Key

Use the AWS managed customer master key for CodePipeline in your account to encrypt the data in the artifact store.

Customer Managed Key

To encrypt the data in the artifact store under an AWS KMS customer managed key, specify the key ID, key ARN, or alias ARN.

Cancel

Next

Enter the pipeline name: MF-AWS-CICD-Pipeline.

We select Default location for the artifact store, meaning we will not reuse the one created for CodeBuild and CodeDeploy previously but we'll use one which is automatically generated by CodePipeline.

Click Next.

Step 1      Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 2      Choose pipeline settings

**Add source stage**

Step 3      Add build stage

Step 4      Add deploy stage

Step 5      Review

## Add source stage

**Source**

**Source provider**  
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

AWS CodeCommit

**Repository name**  
Choose a repository that you have already created where you have pushed your source code.

MF-AWS-CI-CD-SCM

**Branch name**  
Choose a branch of the repository

master

**Change detection options**  
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.

**Amazon CloudWatch Events (recommended)**  
Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs

**AWS CodePipeline**  
Use AWS CodePipeline to check periodically for changes

Cancel      Previous      **Next**

Select the CodeCommit repository and branch, then click Next.

Step 1      Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 2      Choose pipeline settings

**Add source stage**

Step 3      Add build stage

Step 4      Add deploy stage

Step 5      Review

## Add build stage

**Build - optional**

**Build provider**  
This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.

AWS CodeBuild

**Region**  
US East - (N. Virginia)

**Project name**  
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

MF-AWS-CI-CD-Build

**Environment variables - optional**  
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

Add environment variable

Cancel      Previous      Skip build stage      **Next**

Click Next

## Add deploy stage

**Deploy - optional**

**Deploy provider**  
Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

▼

Cancel Previous **Skip deploy stage** Next

Click Skip deploy stage.

Click Create pipeline.

# 5 DEPLOY to TEST stage

## 5.1 DEPLOY to TEST – MF ETS configuration for CodeDeploy

In AWS console, create an IAM role for CodeDeploy access.

You can find more help for these steps at this link:

<https://docs.aws.amazon.com/codedeploy/latest/userguide/getting-started-create-iam-instance-profile.html>

Create an EC2-to-S3-CodeDeploy-microfocus-aws-readonly IAM role with inline policy:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:Get*",  
                "s3>List*"  
            ],  
            "Resource": [  
                "arn:aws:s3:::mf-aws-cicd-artifacts/*",  
                "arn:aws:s3:::aws-codedeploy-us-east-2/*",  
                "arn:aws:s3:::aws-codedeploy-us-east-1/*",  
                "arn:aws:s3:::aws-codedeploy-us-west-1/*",  
                "arn:aws:s3:::aws-codedeploy-us-west-2/*",  
                "arn:aws:s3:::aws-codedeploy-ca-central-1/*",  
                "arn:aws:s3:::aws-codedeploy-eu-west-1/*",  
                "arn:aws:s3:::aws-codedeploy-eu-west-2/*",  
                "arn:aws:s3:::aws-codedeploy-eu-west-3/*",  
                "arn:aws:s3:::aws-codedeploy-eu-central-1/*",  
                "arn:aws:s3:::aws-codedeploy-ap-east-1/*",  
                "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",  
                "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",  
                "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",  
                "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",  
                "arn:aws:s3:::aws-codedeploy-ap-south-1/*",  
                "arn:aws:s3:::aws-codedeploy-sa-east-1/*",  
                "arn:aws:s3:::codepipeline-us-east-2-*",  
                "arn:aws:s3:::codepipeline-us-east-1-*",  
                "arn:aws:s3:::codepipeline-us-west-1-*",  
                "arn:aws:s3:::codepipeline-us-west-2-*",  
                "arn:aws:s3:::codepipeline-ca-central-1-*",  
                "arn:aws:s3:::codepipeline-eu-west-1-*",  
                "arn:aws:s3:::codepipeline-eu-west-2-*",  
                "arn:aws:s3:::codepipeline-eu-west-3-*",  
                "arn:aws:s3:::codepipeline-eu-central-1-*",  
                "arn:aws:s3:::codepipeline-ap-east-1-*",  
                "arn:aws:s3:::codepipeline-ap-northeast-1-*",  
                "arn:aws:s3:::codepipeline-ap-northeast-2-*",  
                "arn:aws:s3:::codepipeline-ap-southeast-1-*",  
                "arn:aws:s3:::codepipeline-ap-southeast-2-*",  
                "arn:aws:s3:::codepipeline-ap-south-1-*",  
                "arn:aws:s3:::codepipeline-sa-east-1-*"  
            ]  
        }  
    ]  
}
```

Create CodeDeploy-to-EC2-microfocus-aws IAM role with the attached AWSCodeDeployRole policy.

## Summary

Role ARN: arn:aws:iam::954254376221:role/CodeDeploy-to-EC2-microfocus-aws

Role description: Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf. | [Edit](#)

Instance Profile ARNs:

Path: /

Creation time: 2020-02-18 13:30 EST

Last activity: Not accessed in the tracking period

Maximum CLI/API session duration: 1 hour [Edit](#)

[Permissions](#) [Trust relationships](#) [Tags](#) [Access Advisor](#) [Revoke sessions](#)

▼ Permissions policies (1 policy applied)

[Attach policies](#)

Policy name	Policy type
AWSCodeDeployRole	AWS managed policy

Copy the Role ARN for this role such as arn:aws:iam::954254376221:role/CodeDeploy-to-EC2-microfocus-aws

For getting access to Micro Focus software, please contact your Micro Focus representative or contact Micro Focus following this link: <https://www.microfocus.com/en-us/contact/contactme>

First you need to retrieve an AMI with Micro Focus Enterprise Test Server (ETS) or deploy the ETS software on an EC2 instance.

Select Micro Focus Enterprise Test Server (ETS) AMI and click Launch.

Under Configure Instance Details, choose the EC2-to-S3-CodeDeploy-microfocus-aws-readonly role.

IAM role [EC2-to-S3-CodeDeploy-microfocus-aws-readonly](#) Create new IAM role

Under Advanced Details User data, enter the following commands to install CodeDeploy agent:

```
<powershell>
Set-ExecutionPolicy RemoteSigned -Force
Import-Module AWSPowerShell
$REGION = (ConvertFrom-Json (Invoke-WebRequest -Uri http://169.254.169.254/latest/dynamic/instance-
identity/document -UseBasicParsing).Content).region
New-Item -Path c:\temp -ItemType "directory" -Force
powershell.exe -Command Read-S3Object -BucketName aws-codedeploy-$REGION -Key latest/codedeploy-agent-
updater.msi -File c:\temp\codedeploy-agent-updater.msi
// Start-Sleep -Seconds 30 *optional
c:\temp\codedeploy-agent-updater.msi /quiet /l c:\temp\host-agent-updater-log.txt
</powershell>
```

▼ Advanced Details

User data  As text  As file  Input is already base64 encoded

```
<powershell>
Set-ExecutionPolicy RemoteSigned -Force
Import-Module AWSPowerShell
$REGION = (ConvertFrom-Json (Invoke-WebRequest -Uri
http://169.254.169.254/latest/dynamic/instance-identity/document -
UseBasicParsing).Content).region
```

Add a tag with Key CodeDeployGroup and Value ETS-MF-AWS-CICD.

## Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.

A copy of a tag can be applied to volumes, instances or both.

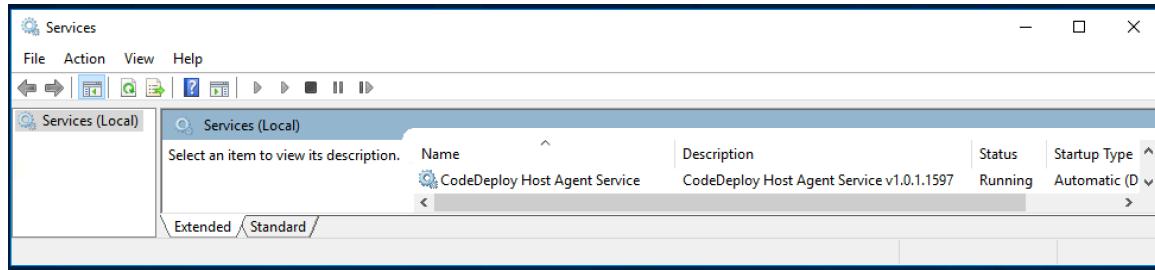
Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key	(128 characters maximum)	Value	(256 characters maximum)	Instances
CodeDeployGroup		ETS-MF-AWS-CICD		<input checked="" type="checkbox"/>

**Add another tag** (Up to 50 tags maximum)

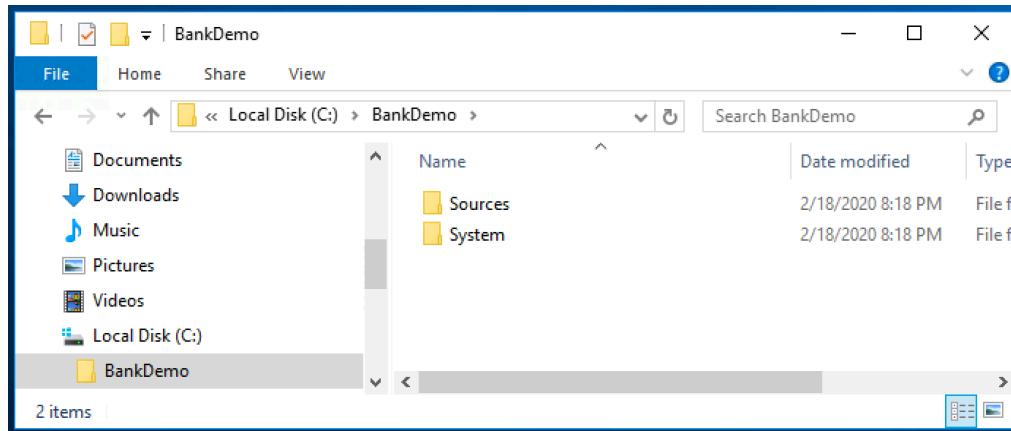
Then Launch the EC2 instance.

Connect to the launched instance and verify the CodeDeploy agent is installed and started:



We're now going to configure a BankDemo test server on ETS.

Zip and copy the BankDemo folder within C:\Users\Public\Documents\Micro Focus\Enterprise Developer\Samples>Mainframe\BankDemo\MFETDUSER from the EDz instance, and extract it on the ETS instance right under the root directory C:\. Be careful not to extract it under C:\BankDemo\BankDemo (folder duplicated).



Verify the C:\BankDemo\System\SysLoadlib folder has the proper letters in upper-case (S and L).

Name	Date modified	Type
catalog	2/19/2020 11:20 PM	File folder
DATA	2/19/2020 11:26 PM	File folder
Logs	2/25/2020 4:16 PM	File folder
RDEF	2/19/2020 11:20 PM	File folder
SysLoadlib	2/19/2020 11:20 PM	File folder

With a browser, go to the Enterprise Server Administration at <http://localhost:86/>

The screenshot shows the Micro Focus Enterprise Server Administration interface. The top navigation bar includes links for Home, Actions (Renew, Export, Import, Delete All), Configure (Options, Security), Display (Statistics, Sessions, Journal), and Help (This Page, Support, Feedback). The main content area displays a list of servers. A message at the top states: "This UI is currently configured to be accessible only from the local machine. Change the TCP on loopback option here and restart the MF Directory Server process to enable remote access." Below this, a table lists servers with columns for Type, Name, Status, Communications Processes, Licenses, Security, and Status Log. One server, named 'ESDEMO', is shown in detail. Its status is 'Stopped' with a note: 'tcp.\*:9000' and '3 Listeners'. The 'Status Log' section shows a message: 'CASST0001I Server manager termination completed successfully 19:59:03'. Other details include '5 Services', '3 Handlers', and '0 Packages'. The status log also notes: '1 minute 15 seconds in "Stopped" state since 19:59:04 02/18/20' and 'Stopped by admin ID "mfuser" using ES ID "mf\_mdsa" under system ID "SYSTEM"'.

Click Import in the left upper corner of Enterprise Server Administration.

On the Import server information page and under Recent directories click the directory for the BANKDEMO server.

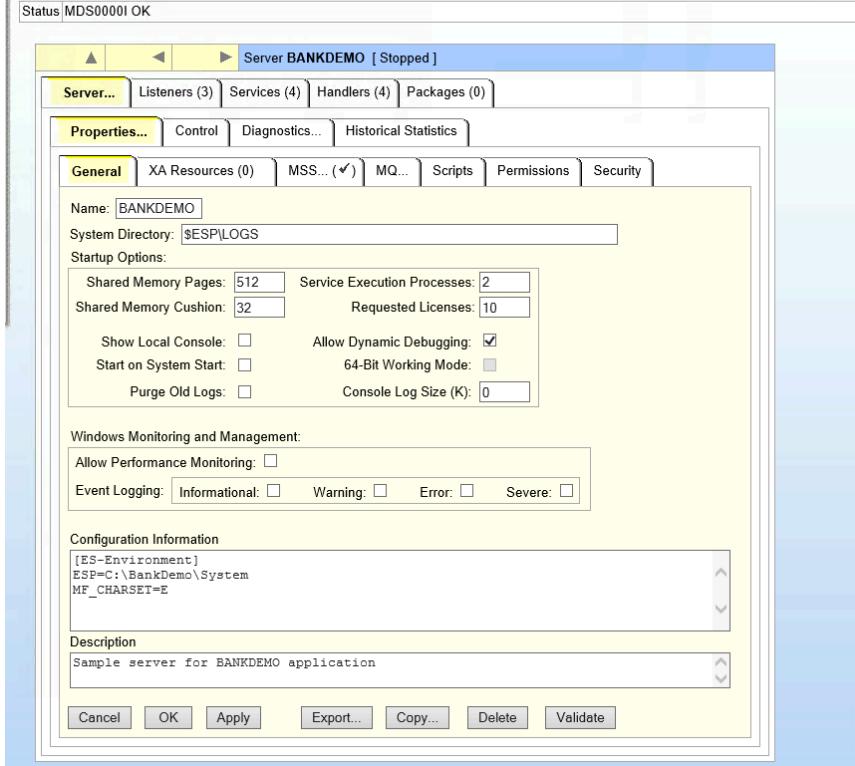
This adds the path to the Selected source directory containing server data to restore field.  
Click Next 3 times and click OK to import the BankDemo server (keep the server in 32-bit).

The system returns to the main Enterprise Server Administration page.

You can see the Bankdemo server appears in the list of servers.

In front of the BANKDEMO server, select Edit...

Enterprise Server Administration > BANKDEMO > Edit  
localhost:86  
⚠ This UI is currently configured to be accessible only from the local machine. Change the TCP on loopback option here and restart the MF Directory Server process to enable remote access.



Under Configuration Information, update the ESP variable path to the actual location on ETS instance:

ESP=C:\BankDemo\System

Click OK to save.

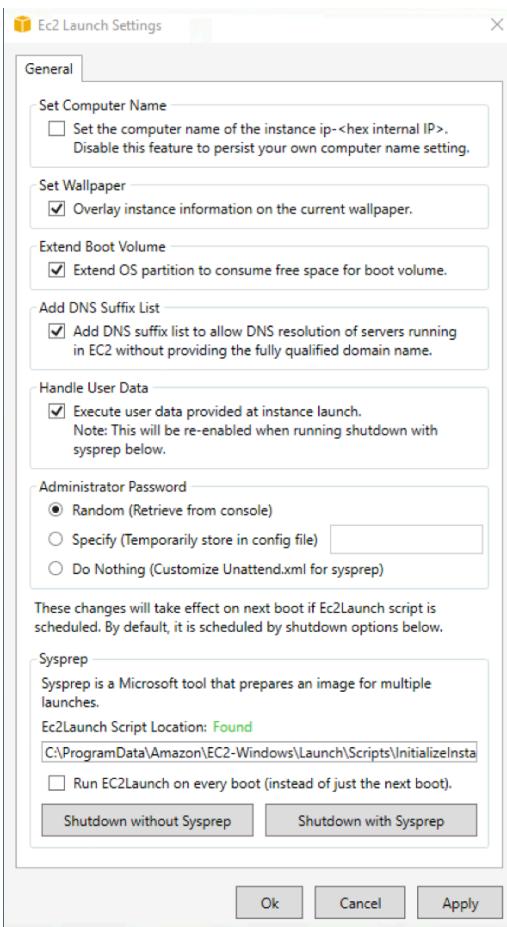
Under the Enterprise Server Administration, click Start for the BANKDEMO server...

	Type	Name	Status	Communications Processes	Licenses	Security	Status Log	Objects
	MFES (MSS)	BANKDEMO	Started	1 tcp:172.31.24.87:51014* (172.31.24.87) ✓ 3 Listeners	10 / 10	Default	Server: CP 1:  MDS3801I Server started successfully 20:27:41 02/18/20  1 minute 8 seconds in "Started" state since 20:27:41 02/18/20 Started by admin ID "mfuser" using ES ID "mf_mds" under system ID "SYSTEM", Server Manager PID 1140	4 Services 4 Handlers 0 Packages

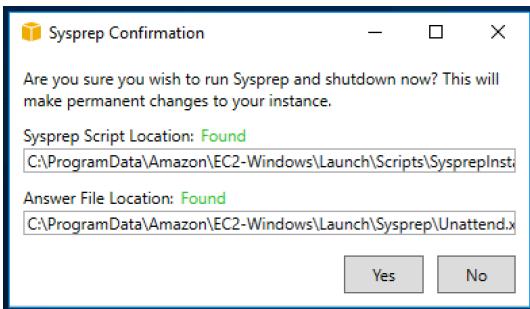
If any server start problem you can check the console log in C:\BankDemo\System\Logs\console.txt.

The BankDemo compiled dll files will go in C:\BankDemo\System\SysLoadlib

Run C:\ProgramData\Amazon\EC2-Windows\Launch\Settings\Ec2LaunchSettings.exe

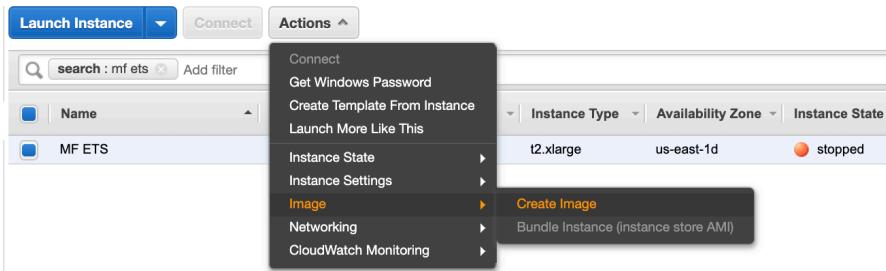


## Click Shutdown with Sysprep



Click Yes.

Once the ETS instance is stopped (not terminated), create an AMI from it: Micro Focus ETS with BankDemo.



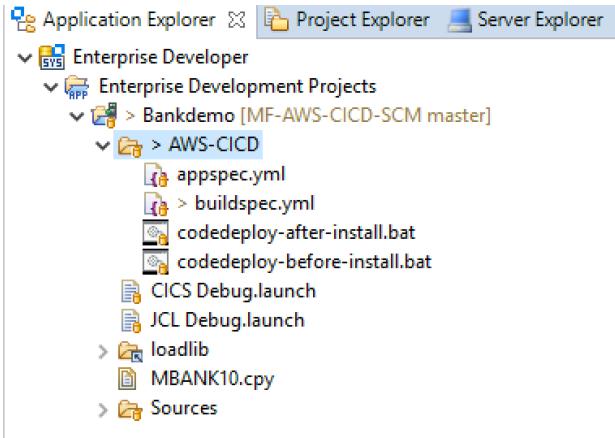
Name	AMI Name
	Micro Focus ETS with BankDemo

This new AMI will be used in case you need to terminate and start new ETS instances with BankDemo server already configured.

Every time you start a new EC2 instance for ETS with BankDemo, verify you have the role EC2-to-S3-CodeDeploy-microfocus-aws-readonly configured under Configure Instance Details. Also verify you have a tag assigned with Key CodeDeployGroup and Value ETS-MF-AWS-CICD.

## 5.2 DEPLOY to TEST – AWS CodeDeploy configuration

In the EDz Bankdemo project, under AWS-CICD folder, create three files: appspec.yml, codedeploy-before-install.bat, codedeploy-after-install.bat



appspec.yml content:

```

version: 0.0
os: windows
files:
  - source: .\dll
    destination: .\dll-staging
hooks:
  BeforeInstall:
    - location: \codedeploy-before-install.bat
      timeout: 120
  AfterInstall:
    - location: \codedeploy-after-install.bat
      timeout: 120
  
```

The application specification file (AppSpec file) is a YAML-formatted or JSON-formatted file used by CodeDeploy to manage a deployment. Documentation about it is available here:

[https://docs.amazonaws.cn/en\\_us/codedeploy/latest/userguide/reference-appspec-file.html](https://docs.amazonaws.cn/en_us/codedeploy/latest/userguide/reference-appspec-file.html)

codedeploy-before-install.bat content:

```
C:\\"Program Files (x86)"\"Micro Focus"\\"Enterprise Test Server"\bin\casstop /lBANKDEMO  
mkdir .\dll-staging  
ping 127.0.0.1 -n 30 -w 1000 > NUL
```

codedeploy-after-install.bat content:

```
copy /b/v/y .\dll-staging\* C:\BankDemo\System\SysLoadlib  
C:\\"Program Files (x86)"\"Micro Focus"\\"Enterprise Test Server"\bin\casstart /rBANKDEMO  
ping 127.0.0.1 -n 30 -w 1000 > NUL
```

Then Commit and Push these file changes making sure the new files are staged.

Then make a Build in CodeBuild to update the artifacts in S3.

From the AWS console, go to Developer Tools, CodeDeploy, then Applications.

Click Create application.

For the Application name, enter: ETS-for-MF-AWS-CICD.

For the Compute platform, select EC2/On-premises.

Developer Tools > CodeDeploy > Applications > Create application

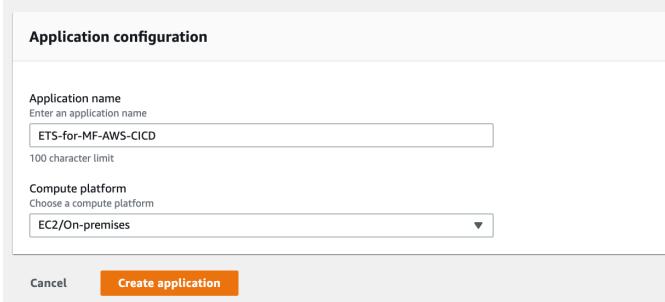
Create application

Application configuration

Application name  
Enter an application name  
ETS-for-MF-AWS-CICD  
100 character limit

Compute platform  
Choose a compute platform  
EC2/On-premises

Cancel Create application



Click Create application.

Under the ETS-for-MF-AWS-CICD, select Create deployment group.

Enter deployment group name: ETS-EC2-instances-MF-AWS-CICD

Enter service role ARN previously created such as arn:aws:iam::954254376221:role/CodeDeploy-to-EC2-microfocus-aws

Deployment type is In-place.

Environment configuration is Amazon EC2 instances only.

For the tag, the Key is CodeDeployGroup and Value is ETS-EC2-instances-MF-AWS-CICD.

**Deployment group name**

Enter a deployment group name  
  
100 character limit

**Service role**

Enter a service role  
Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.

X

**Deployment type**

Choose how to deploy your application

**In-place**  
Updates the instances in the deployment group with the latest application revisions. During a deployment, each instance will be briefly taken offline for its update

**Blue/green**  
Replaces the instances in the deployment group with new instances and deploys the latest application revision to them. After instances in the replacement environment are registered with a load balancer, instances from the original environment are deregistered and can be terminated.

**Environment configuration**

Select any combination of Amazon EC2 Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add to this deployment

Amazon EC2 Auto Scaling groups

Amazon EC2 instances  
1 unique matched instance. [Click here for details](#)

You can add up to three groups of tags for EC2 instances to this deployment group.  
**One tag group:** Any instance identified by the tag group will be deployed to.  
**Multiple tag groups:** Only instances identified by all the tag groups will be deployed to.

Tag group 1

Key	Value - optional
<input type="text" value="CodeDeployGroup"/> <span style="border: 1px solid #ccc; padding: 2px;">X</span>	<input type="text" value="ETS-EC2-instances-MF-AWS-CICD"/> <span style="border: 1px solid #ccc; padding: 2px;">X</span>

Remove tag

Add tag

+ Add tag group

On-premises instances

**Matching instances**  
1 unique matched instance. [Click here for details](#)

You can verify it found the ETS instance accordingly:

**Matching instances**  
1 unique matched instance. [Click here for details](#)

Deployment settings is CodeDeployDefault.AllAtOnce.

Deselect Load balancer which are not used here.

**Deployment settings**

**Deployment configuration**  
Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application is deployed and the success or failure conditions for a deployment.

CodeDeployDefault.AllAtOnce ▾ or [Create deployment configuration](#)

**Load balancer**

Select a load balancer to manage incoming traffic during the deployment process. The load balancer blocks traffic from each instance while it's being deployed to and allows traffic to it again after the deployment succeeds.

Enable load balancing

Then click Create Deployment group.

Developer Tools > CodeDeploy > Applications > ETS-for-MF-AWS-CICD > ETS-EC2-instances-MF-AWS-CICD

**ETS-EC2-instances-MF-AWS-CICD** [Edit](#) [Delete](#) [Create deployment](#)

Deployment group details		
Deployment group name ETS-EC2-instances-MF-AWS-CICD	Application name <a href="#">ETS-for-MF-AWS-CICD</a>	Compute platform EC2/On-premises
Deployment type In-place	Service role ARN arn:aws:iam::954254376221:role/CodeDeploy-to-EC2-microfocus-aws	Deployment configuration <a href="#">CodeDeployDefault.AllAtOnce</a>
Rollback enabled False		

Under CodeDeploy -> Applications -> Application -> ETS-for-MF-AWS-CICD select the Deployments tab, then click Create deployment.

Developer Tools > CodeDeploy > Applications > ETS-for-MF-AWS-CICD

**ETS-for-MF-AWS-CICD** [Notify](#) [Delete application](#)

Application details	
Name ETS-for-MF-AWS-CICD	Compute platform EC2/On-premises

[Deployments](#) [Deployment groups](#) [Revisions](#)

**Application deployment history** [C](#) [View details](#) [Actions](#) [Copy deployment](#) [Retry deployment](#) [Create deployment](#)

## Create deployment

**Deployment settings**

Application  
ETS-for-MF-AWS-CICD

Deployment group

Compute platform  
EC2/On-premises

Deployment type  
In-place

Revision type  
 My application is stored in Amazon S3  My application is stored in GitHub

Revision location  
Copy and paste the Amazon S3 bucket where your revision is stored  
   
s3://bucket-name/folder/object.[zip|tar|tgz]

Revision file type

For Deployment group, choose: ETS-EC2-instances-MF-AWS-CICD

For Revision location, enter: s3://mf-aws-cicd-artifacts/MF-AWS-CICD-Build/deploy.zip

Revision file type is .zip

We don't need to override the content because the codedeploy-after-install.bat script takes care of it.

Then click Create deployment.

d-1XEZEEVO2

[Copy deployment](#)[Retry deployment](#)

**Deployment status**

Installing application on your instances	1 of 1 instances updated	<span>Success</span> Succeeded
--	--------------------------	--------------------------------

**Deployment details**

Application <a href="#">ETS-for-MF-AWS-CICD</a>	Deployment ID d-1XEZEEVO2	Status <span>Success</span> Succeeded
Deployment configuration <a href="#">CodeDeployDefault.AllAtOnce</a>	Deployment group <a href="#">ETS-EC2-instances-MF-AWS-CICD</a>	Initiated by User action
Deployment description -		

**Revision details**

Revision location <a href="#">s3://mf-aws-cicd-artifacts/MF-AWS-CICD-Build/deploy.zip</a>	Revision created 10 hours ago	Revision description Application revision registered by Deployment ID: d-PDT76QJ02
--	----------------------------------	---

**Deployment lifecycle events**

Instance ID	Duration	Status	Most recent event	Events	Start time	End time
i-04bb3a3a977b43d4f	7 seconds	<span>Success</span> Succeeded	ValidateService	<a href="#">View events</a>	Feb 19, 2020 9:48 AM (UTC-5:00)	Feb 19, 2020 9:48 AM (UTC-5:00)

Clicking View events, you can see the details of the deployment.

**Revision details**

Revision location <a href="#">s3://mf-aws-cicd-artifacts/MF-AWS-CICD-Build/deploy.zip</a>	Revision created 10 hours ago	Revision description Application revision registered by Deployment ID: d-PDT76QJ02
--	----------------------------------	---

**Event details**

Event	Duration	Status	Error code	Start time	End time
ApplicationStop	less than one second	<span>Success</span> Succeeded	-	Feb 19, 2020 9:48 AM (UTC-5:00)	Feb 19, 2020 9:48 AM (UTC-5:00)
DownloadBundle	less than one second	<span>Success</span> Succeeded	-	Feb 19, 2020 9:48 AM (UTC-5:00)	Feb 19, 2020 9:48 AM (UTC-5:00)
BeforeInstall	less than one second	<span>Success</span> Succeeded	-	Feb 19, 2020 9:48 AM (UTC-5:00)	Feb 19, 2020 9:48 AM (UTC-5:00)
Install	less than one second	<span>Success</span> Succeeded	-	Feb 19, 2020 9:48 AM (UTC-5:00)	Feb 19, 2020 9:48 AM (UTC-5:00)
AfterInstall	1 second	<span>Success</span> Succeeded	-	Feb 19, 2020 9:48 AM (UTC-5:00)	Feb 19, 2020 9:48 AM (UTC-5:00)
ApplicationStart	0 seconds	<span>Success</span> Succeeded	-	Feb 19, 2020 9:48 AM (UTC-5:00)	Feb 19, 2020 9:48 AM (UTC-5:00)
ValidateService	less than one second	<span>Success</span> Succeeded	-	Feb 19, 2020 9:48 AM (UTC-5:00)	Feb 19, 2020 9:48 AM (UTC-5:00)

Then you can verify that the dll files have been updated in the ETS C:\BankDemo\System\SysLoadlib folder.

If you need to troubleshoot the CodeDeploy agent, the downloaded files and logs are under C:\ProgramData\Amazon\CodeDeploy.

## 5.3 DEPLOY to TEST – AWS CodePipeline configuration

In CodePipeline, select the MF-AWS-CICD-Pipeline and click Edit.

Click Add stage after the Build stage.

Add stage

Stage name

Deploy to Test

No more than 100 characters

Cancel Add stage

Enter stage name: Deploy-to-Test

Click Add stage.

For this new stage, click Add action group.

Enter action name: Deploy-to-Test

Select Deploy provider AWS CodeDeploy

Select Input artifacts: BuildArtifact

Select Application name ETS-for-MF-AWS-CICD

Select Deployment group ETS-EC2-instances-MF-AWS-CICD

Edit action

Action name  
Choose a name for your action

Deploy-to-Test

No more than 100 characters

Action provider

AWS CodeDeploy

Region

US East - (N. Virginia)

Input artifacts  
Choose an input artifact for this action. [Learn more](#)

BuildArtifact

No more than 100 characters

Application name  
Choose an application that you have already created in the AWS CodeDeploy console. Or create an application in the AWS CodeDeploy console and then return to this task.

ETS-for-MF-AWS-CICD

Deployment group  
Choose a deployment group that you have already created in the AWS CodeDeploy console. Or create a deployment group in the AWS CodeDeploy console and then return to this task.

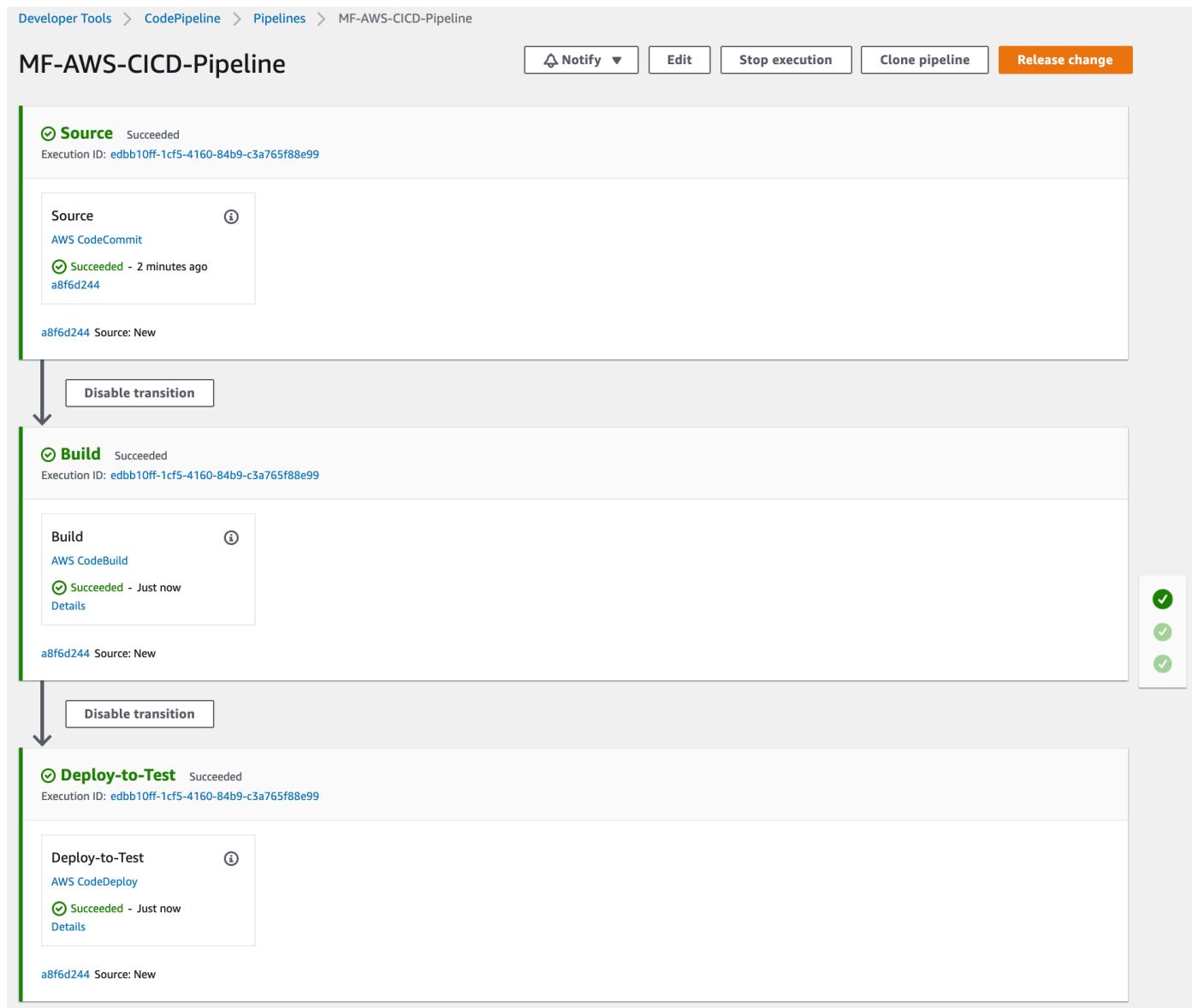
ETS-EC2-instances-MF-AWS-CICD

Variable namespace - *optional*  
Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. [Learn more](#)

Click Done

Save the modified pipeline.

As soon as there is a code change pushed into CodeCommit, the pipeline executes.



# 6 TEST stage

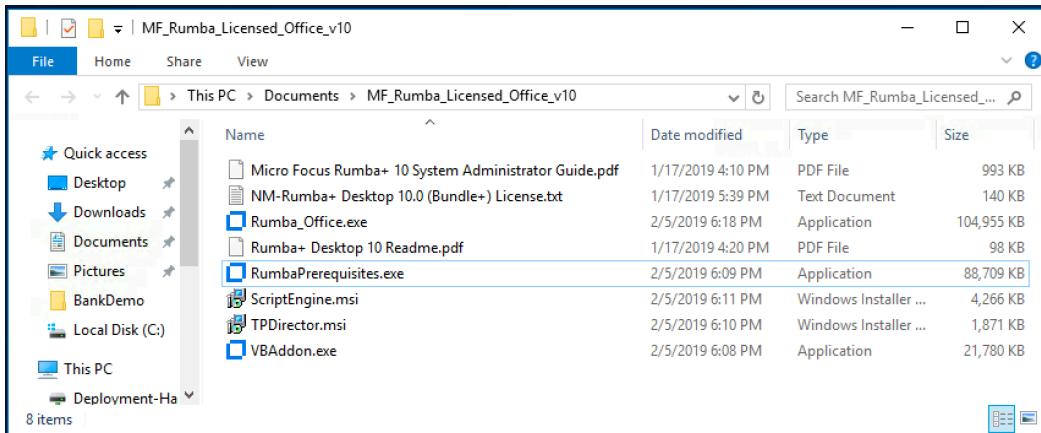
## 6.1 TEST – MF ETS configuration for automated tests

Tests are launched from this batch script: C:\BankDemo\Test\BankDemo-Tests.bat which will be called by CodePipeline, via Lambda and AWS Systems Manager (SSM).

For getting access to Micro Focus software, please contact your Micro Focus representative or contact Micro Focus following this link: <https://www.microfocus.com/en-us/contact/contactme>

First you need to retrieve Micro Focus Rumba+ Office and VB Add-on software installer.

If not already installed on ETS instance, install Rumba prerequisites, Rumba+ Desktop Office, and VB Add-on.



Reboot ETS instance from AWS console.

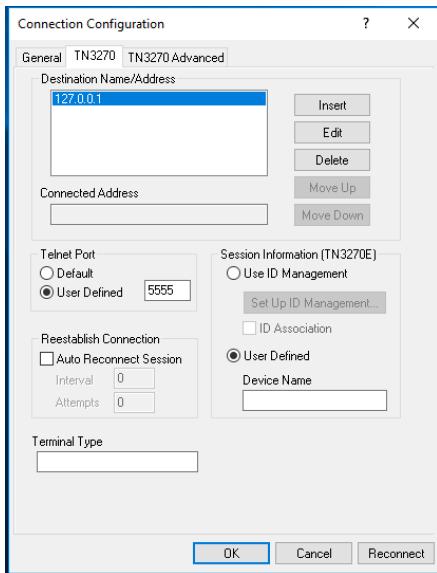
Start the BankDemo server:

```
C:\\"Program Files (x86)"\"Micro Focus"\\"Enterprise Test Server"\bin\casstart /rBANKDEMO
```

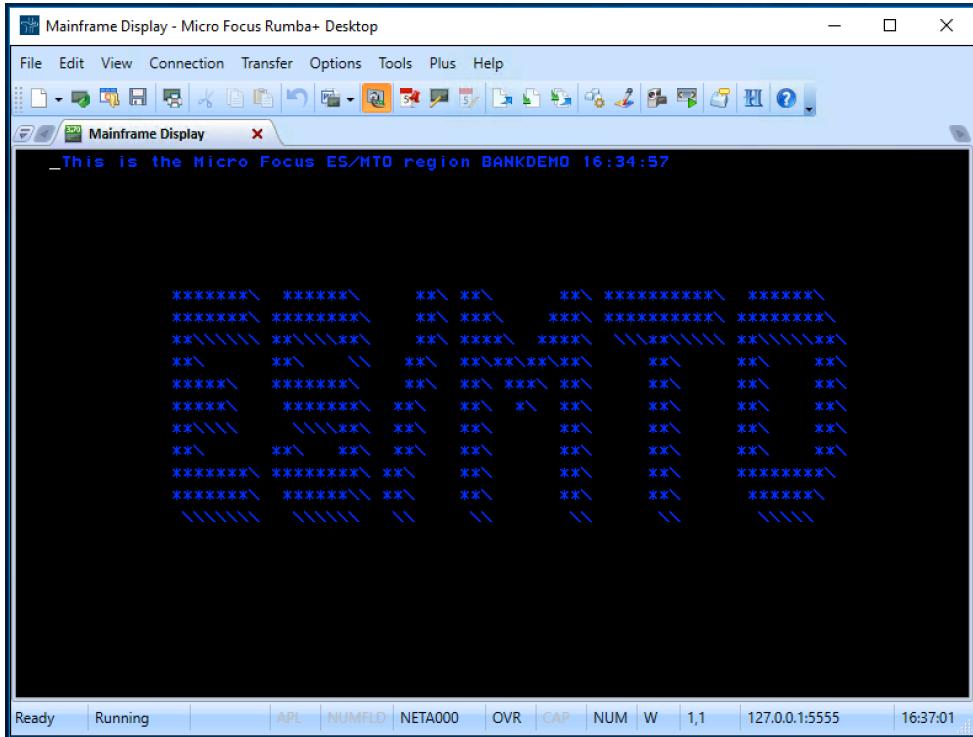
If any server start problem you can check the console log in C:\BankDemo\System\Logs\console.txt.

Start Rumba+ Desktop.

Configure a TN3270 to IP address 127.0.0.1 and port 5555.

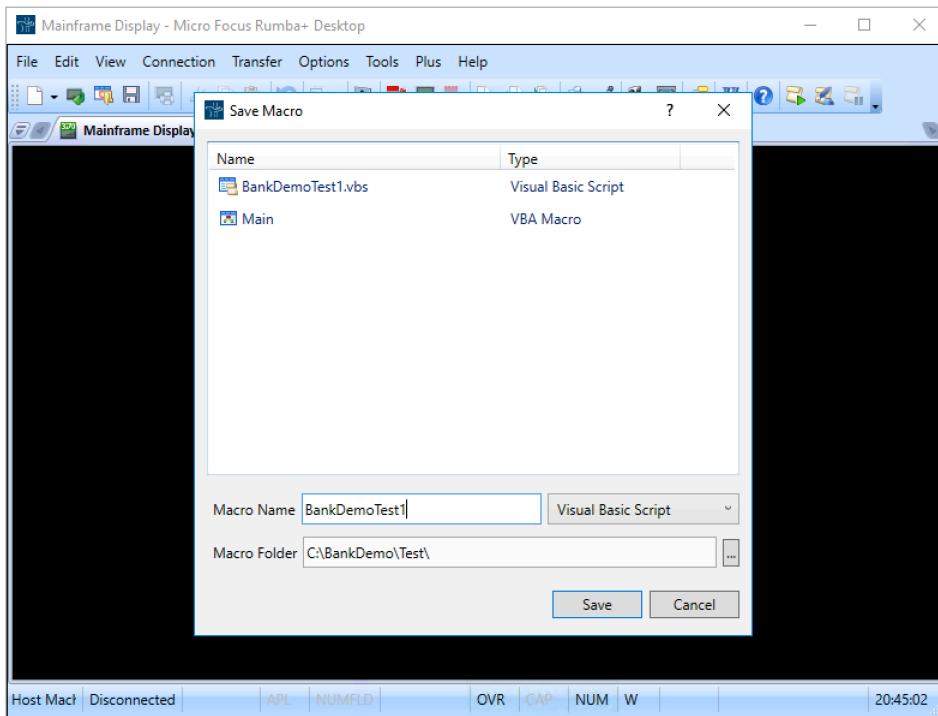


Then connect with this connection.



Verify it connects successfully and that you can run manual tests.

If you need to record Rumba test script, record a macro and then save it under Visual Basic Script format in order to save it to a folder.



## Create a test script in C:\BankDemo\Test\BankDemoTest1.vbs

```

WScript.Stdout.WriteLine "Connecting Rumba session..."
sessType = Conn_3270
Set app = CreateObject("MicroFocus.Rumba")
If app.GetSessionType(app.ActiveSessionID) = sessType Then
    Set session = app.GetSession(app.ActiveSessionID)
Else
    sessID = app.CreateSession(sessType)
    Set session = app.GetSession(sessID)
End if

session.HostName = "127.0.0.1"
session.Port = 5555
session.Connect()
WScript.Stdout.WriteLine "Session connected."

WScript.Stdout.WriteLine "Interacting with screens..."
WaitScreen "This is the Micro Focus ES/MTO region", DefaultConnectionTimeout, 1, 2, SearchOnlyAt, False, Empty, Empty
session.SendKey "Clear"
WaitScreenTimeout DefaultScreenTimeout
session.TypeText "bank"
session.SendKey "Enter"
WaitScreen "User id.....:", DefaultScreenWidthTimeout, 10, 30, SearchOnlyAt, False, 10, 44
session.TypeText "b0001v"
session.SendKey "Enter"
WaitScreen "*****", DefaultScreenWidthTimeout, 2, 17, SearchOnlyAt, False, 8, 4
"***MISMATCH*****", DefaultScreenWidthTimeout, 2, 17, SearchOnlyAt, False, 8, 4
session.TypeText "/"
session.SendKey "Enter"
WaitScreen "450061494 ", DefaultScreenWidthTimeout, 11, 8, SearchOnlyAt, False, 11, 3
session.TypeText "/"
session.SendKey "Enter"
WaitScreen "Scrn:", DefaultScreenWidthTimeout, 1, 2, SearchOnlyAt, False, Empty, Empty
session.SendKey "PF4"
WaitScreen "450061494 ", DefaultScreenWidthTimeout, 11, 8, SearchOnlyAt, False, Empty, Empty
session.SendKey "Tab"
session.TypeText "/"
session.SendKey "Enter"
WaitScreen "Scrn:", DefaultScreenWidthTimeout, 1, 2, SearchOnlyAt, False, Empty, Empty
session.SendKey "PF4"
WaitScreen "450061494 ", DefaultScreenWidthTimeout, 11, 8, SearchOnlyAt, False, Empty, Empty
session.SendKey "PF4"
WaitScreen "*****", DefaultScreenWidthTimeout, 2, 17, SearchOnlyAt, False, Empty,
Empty
session.SendKey "Tab"
session.SendKey "Tab"
session.SendKey "Tab"
session.TypeText "/"
session.SendKey "Enter"
WaitScreen "The amount you would like to borrow...:", DefaultScreenWidthTimeout, 8, 6, SearchOnlyAt, False, 8, 46
session.TypeText "10000"
session.SendKey "Tab"
session.TypeText "4.25"
session.SendKey "Tab"

```

```

session.TypeText "24"
session.SendKey "Enter"
WaitScreen "The amount you would like to borrow...:", DefaultScreenDataTimeout, 8, 6, SearchOnlyAt, False, Empty, Empty
session.SendKey "PF4"
WaitScreen "*****", DefaultScreenDataTimeout, 2, 17, SearchOnlyAt, False, Empty, Empty
Empty
session.SendKey "PF4"
WaitScreen "User id.....:", DefaultScreenDataTimeout, 10, 30, SearchOnlyAt, False, Empty, Empty
session.SendKey "PF3"
WaitScreenTimeout DefaultScreenTimeout
WScript.Stdout.WriteLine "Screen interactions completed."

session.Disconnect()
WScript.Stdout.WriteLine "Session disconnected."

lResult = CreateObject("WScript.Shell").Run("taskkill /f /im RumbaPage.exe", 0, True)
WScript.Stdout.WriteLine "Rumba process killed."
WScript.Stdout.WriteLine "BANKDEMO TESTS COMPLETED WITH SUCCESS"

Const DefaultScreenTimeout = 3000
Const DefaultScreenDataTimeout = 10000
Const DefaultConnectionTimeout = 10000

Const SearchAnywhere = 0
Const SearchStartingAt = 1
Const SearchOnlyAt = 2

Const ErrorCodeScreenTimeout = 1
Const ErrorCodeSessionDisconnected = 2
Const ErrorCodeHostBusy = 3

Const Conn_3270 = 1
Const Conn_5250 = 2
Const Conn_VAX = 3
Const Conn_Other = 4

Function GetScreenPosition(row, column)
    Dim rows
    Dim columns
    session.GetScreenSize rows, columns
    GetScreenPosition = (columns*(row-1)) + column
End Function

Function ScreenMatch(textToSearch, row, column, searchCriteria, ignoreCase)
    screenPosition = 1
    If (searchCriteria = SearchStartingAt Or searchCriteria = SearchOnlyAt) And Not IsEmpty(row) And Not IsEmpty(column) Then
        screenPosition = GetScreenPosition(row, column)
    End if

    If (searchCriteria = SearchStartingAt Or searchCriteria = SearchOnlyAt) And (IsEmpty(row) Or IsEmpty(column)) Then
        Dim currentRow
        Dim currentColumn
        session.GetCursorPosition currentRow, currentColumn
        If (searchCriteria = SearchStartingAt) Then currentColumn = 1
        screenPosition = GetScreenPosition(currentRow, currentColumn)
    End if

    textToSearchTemp = textToSearch
    screenTextTemp = session.ScreenText
    If ignoreCase = True Then
        textToSearchTemp = UCASE(textToSearchTemp)
        screenTextTemp = UCASE(screenTextTemp)
    End if

    If searchCriteria = SearchOnlyAt Then
        ScreenMatch = Mid(screenTextTemp, screenPosition, Len(textToSearchTemp)) = textToSearchTemp
    Else
        ScreenMatch = InStr(screenPosition, screenTextTemp, textToSearchTemp) <> 0
    End if
End Function

Sub WaitScreen(textToSearch, timeout, row, column, searchCriteria, ignoreCase, cursorPosRowToWait, cursorPosColumnToWait)
    Dim timePassed
    Dim screenFound
    Dim cursorPosMatch
    Dim cursorRow, cursorColumn
    timePassed = 0

    Do
        WScript.Sleep 100
        timePassed = timePassed + 100
        screenFound = ScreenMatch(textToSearch, row, column, searchCriteria, ignoreCase)
        If IsEmpty(cursorPosRowToWait) Or IsEmpty(cursorPosColumnToWait) Then
            cursorPosMatch = True
        Else
            session.GetCursorPosition cursorRow, cursorColumn
            cursorPosMatch = (cursorRow = cursorPosRowToWait And cursorColumn = cursorPosColumnToWait)
        End If
    Loop Until (session.HostReady = True And screenFound = True And cursorPosMatch = True) Or timePassed >= timeout

    If session.Connected = False Then Call Quit(ErrorCodeSessionDisconnected)
    If session.HostReady = False Then Call Quit(ErrorCodeHostBusy)
    If screenFound = False Then Call Quit(ErrorCodeScreenTimeout)
End Sub

Sub WaitScreenTimeout(timeout)
    Dim timePassed

```

```

timePassed = 0
Do
    WScript.Sleep 100
    timePassed = timePassed + 100
Loop Until timePassed >= timeout

If session.Connected = False Then Call Quit(ErrorCodeSessionDisconnected)
If session.HostReady = False Then Call Quit(ErrorCodeHostBusy)
End Sub

Function PromptForHiddenText(prompt, caption)
    Set objHiddenText = CreateObject( "MicroFocus.HiddenInput" )
    txt = objHiddenText.GetInput(prompt, caption)
    If txt = Empty Then WScript.Quit
    PromptForHiddenText = txt
End Function

Sub Quit(ErrorCode)
    If ErrorCode = ErrorCodeScreenTimeout Then WScript.StdErr.Write "ERROR - Screen timeout or screen mismatch."
    If ErrorCode = ErrorCodeSessionDisconnected Then WScript.StdErr.Write "ERROR - Session disconnected."
    If ErrorCode = ErrorCodeHostBusy Then WScript.StdErr.Write "ERROR - Host busy."
    WScript.Quit
End Sub

```

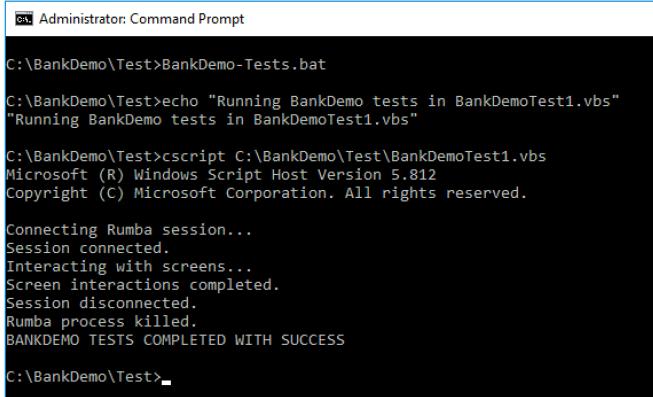
Create a batch file calling this script in C:\BankDemo\Test\BankDemo-Tests.bat

```

echo "Running BankDemo tests in BankDemoTest1.vbs"
cscript C:\BankDemo\Test\BankDemoTest1.vbs

```

Verify the test script runs successfully by running the BankDemo-Tests.bat file.



```

Administrator: Command Prompt
C:\BankDemo\Test>BankDemo-Tests.bat
C:\BankDemo\Test>echo "Running BankDemo tests in BankDemoTest1.vbs"
"Running BankDemo tests in BankDemoTest1.vbs"
C:\BankDemo\Test>cscript C:\BankDemo\Test\BankDemoTest1.vbs
Microsoft (R) Windows Script Host Version 5.812
Copyright (C) Microsoft Corporation. All rights reserved.

Connecting Rumba session...
Session connected.
Interacting with screens...
Screen interactions completed.
Session disconnected.
Rumba process killed.
BANKDEMO TESTS COMPLETED WITH SUCCESS

C:\BankDemo\Test>_

```

## 6.2 TEST – MF ETS configuration for SSM

AWS Lambda uses AWS Systems Manager (SSM) to launch the batch script on the ETS server. Hence we configure SSM first.

Create an IAM role named EC2-to-S3-CodeDeploy-SSM which contains an inline policy with this JSON:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3>List*"
            ],
            "Resource": [
                "arn:aws:s3:::mf-aws-cicd-artifacts/*",

```

```

    "arn:aws:s3:::aws-codedeploy-us-east-2/*",
    "arn:aws:s3:::aws-codedeploy-us-east-1/*",
    "arn:aws:s3:::aws-codedeploy-us-west-1/*",
    "arn:aws:s3:::aws-codedeploy-us-west-2/*",
    "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
    "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
    "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
    "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
    "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
    "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
    "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
    "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
    "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
    "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
    "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
    "arn:aws:s3:::aws-codedeploy-sa-east-1/*",
    "arn:aws:s3:::codepipeline-us-east-2-*",
    "arn:aws:s3:::codepipeline-us-east-1-*",
    "arn:aws:s3:::codepipeline-us-west-1-*",
    "arn:aws:s3:::codepipeline-us-west-2-*",
    "arn:aws:s3:::codepipeline-ca-central-1-*",
    "arn:aws:s3:::codepipeline-eu-west-1-*",
    "arn:aws:s3:::codepipeline-eu-west-2-*",
    "arn:aws:s3:::codepipeline-eu-west-3-*",
    "arn:aws:s3:::codepipeline-eu-central-1-*",
    "arn:aws:s3:::codepipeline-ap-east-1-*",
    "arn:aws:s3:::codepipeline-ap-northeast-1-*",
    "arn:aws:s3:::codepipeline-ap-northeast-2-*",
    "arn:aws:s3:::codepipeline-ap-southeast-1-*",
    "arn:aws:s3:::codepipeline-ap-southeast-2-*",
    "arn:aws:s3:::codepipeline-ap-south-1-*",
    "arn:aws:s3:::codepipeline-sa-east-1-*"
]
}
]
}
}

```

Also attach the **AmazonSSMManagedInstanceCore** managed policy to this role.

[Roles](#) > EC2-to-S3-CodeDeploy-SSM

## Summary

Role ARN	arn:aws:iam::954254376221:role/EC2-to-S3-CodeDeploy-SSM
Role description	Allows EC2 instances to call AWS services on your behalf.   <a href="#">Edit</a>
Instance Profile ARNs	arn:aws:iam::954254376221:instance-profile/EC2-to-S3-CodeDeploy-SSM
Path	/
Creation time	2020-02-22 20:16 EST
Last activity	Not accessed in the tracking period
Maximum CLI/API session duration	1 hour <a href="#">Edit</a>

[Permissions](#) [Trust relationships](#) [Tags](#) [Access Advisor](#) [Revoke sessions](#)

▼ Permissions policies (2 policies applied)

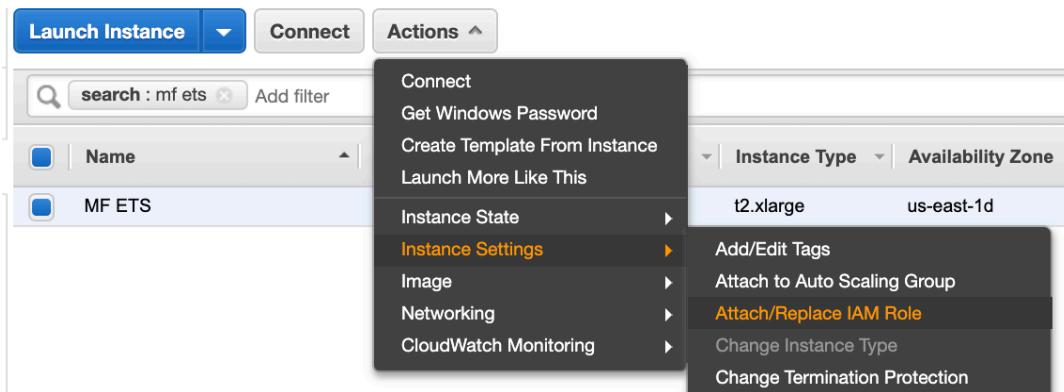
[Attach policies](#)

Policy name	Policy type	AWS managed	Inline policy
AmazonSSMManagedInstanceCore	AWS managed	AWS managed	Inline policy
EC2-to-S3-CodeDeploy-policy	Inline policy	Inline policy	Inline policy

Check if the SSM agent is running on the ETS instance by checking if you see activity in the agent logs in C:\ProgramData\Amazon\SSM\Logs. If no log, follow the instructions to install the SSM agent on Windows:

<https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-install-win.html>

Either launch the ETS instance and attach the role EC2-to-S3-CodeDeploy-SSM, or change the IAM role for the running instance under Instance Settings -> Attach/Replace IAM Role, selecting EC2-to-S3-CodeDeploy-SSM and Apply.



Verify the ETS instance has the tag with key CodeDeployGroup and value ETS-EC2-instances-MF-AWS-CICD.

Stop and Start the ETS instance with the SSM agent.

Verify that the ETS instance shows up under AWS Systems Manager => Instances and Nodes => Managed Instances:

Managed instances			
Instance ID	Name	Ping status	Platform type
i-044b6a7e09129ad64	MF EDz	Online	Windows
i-065db97c182b28445	MF ETS	Online	Windows

Test the SSM configuration with such AWS CLI command:

```
aws ssm send-command --document-name "AWS-RunPowerShellScript" --document-version "1" --targets '[{"Key": "tag:CodeDeployGroup", "Values": ["ETS-EC2-instances-MF-AWS-CICD"]}]' --parameters '{"commands": ["echo Test"], "workingDirectory": [""], "executionTimeout": ["3600"]}' --timeout-seconds 600 --max-concurrency "50" --max-errors "0" --cloud-watch-output-config '{"CloudWatchOutputEnabled":true}' --region us-east-1
```

You can verify the successful completion of this command on the instance via AWS Systems Manager => Run Command => Command history, and then selection the command that was run and the instance.

## Output on i-065db97c182b28445

### Step 1 - Command description and status

Status	Detailed Status	Response code
<span style="color: green;">Success</span>	<span style="color: green;">Success</span>	0

### ▼ Step 1 - Output CloudWatch logs

The command output displays a maximum of 2500 characters. You can view the complete command output in [CloudWatch logs](#).

Test

## 6.3 TEST – AWS Lambda configuration

We now create a Lambda function to execute a command on an EC2 instance.

In IAM, create an IAM role named Lambda-SSM-EC2-Logs which contains an inline policy with this JSON:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:SendCommand",
                "ssm:GetCommandInvocation",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "codepipeline:PutJobSuccessResult",
                "codepipeline:PutJobFailureResult",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

## Summary

[Delete role](#)

<b>Role ARN</b>	arn:aws:iam::954254376221:role/Lambda-SSM-EC2-Logs <a href="#">Copy</a>
<b>Role description</b>	Allows Lambda functions to call AWS services on your behalf. <a href="#">Edit</a>
<b>Instance Profile ARNs</b>	<a href="#">Copy</a>
<b>Path</b>	/
<b>Creation time</b>	2020-02-24 12:57 EST
<b>Last activity</b>	Not accessed in the tracking period
<b>Maximum CLI/API session duration</b>	1 hour <a href="#">Edit</a>

[Permissions](#) [Trust relationships](#) [Tags](#) [Access Advisor](#) [Revoke sessions](#)
▼ Permissions policies (1 policy applied)
[Attach policies](#)[+ Add inline policy](#)

Policy name	Policy type	X
Lambda-SSM-EC2-Logs-policy	Inline policy	X
<div style="border: 1px solid #ccc; padding: 10px;"> <div style="display: flex; justify-content: space-between;"> <span>Policy summary</span> <span>{ JSON</span> <span>Edit policy</span> <span>Simulate policy</span> </div> <pre> 3  "Statement": [ 4    { 5      "Effect": "Allow", 6      "Action": [ 7        "ssm:SendCommand", 8        "ssm:GetCommandInvocation", 9        "ec2:DescribeInstances", 10       "ec2:DescribeInstanceStatus", 11       "codepipeline:PutJobSuccessResult", 12       "codepipeline:PutJobFailureResult", 13       "logs&gt;CreateLogGroup", 14       "logs&gt;CreateLogStream", 15       "logs:PutLogEvents" 16     ], 17     "Resource": "*" </pre> </div>		
<span>▶ Permissions boundary (not set)</span>		

In Lambda, create a Lambda function named CodePipeline-Lambda-SSM-EC2-RunPowerShellScript.

Select Runtime Node.js 12.x.

Select the Lambda-SSM-EC2-Logs existing role.

## Create function Info

Choose one of the following options to create your function.

### Author from scratch

Start with a simple Hello World example.



### Use a blueprint

Build a Lambda application from sample code and configuration presets for common use cases.



### Browse serverless app repository

Deploy a sample Lambda application from the AWS Serverless Application Repository.



## Basic information

### Function name

Enter a name that describes the purpose of your function.

**CodePipeline-Lambda-SSM-EC2-RunPowerShellScript**

Use only letters, numbers, hyphens, or underscores with no spaces.

### Runtime Info

Choose the language to use to write your function.

Node.js 12.x

### Permissions Info

Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

#### ▼ Choose or create an execution role

##### Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

- Create a new role with basic Lambda permissions
- Use an existing role
- Create a new role from AWS policy templates

##### Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

Lambda-SSM-EC2-Logs



[View the Lambda-SSM-EC2-Logs role](#) on the IAM console.

Cancel

Create function

Click Create function.

Enter code inline index.js:

```
const AWS = require('aws-sdk');
const ssm = new AWS.SSM();
const ec2 = new AWS.EC2();
const waitInterval = 1
const timeoutSSM = 300
var jobId
var jobUserParameters

exports.handler = async (event, context) => {
  const instanceIds = [];
  var instanceTagKey = 'CodeDeployGroup'
  var instanceTagValue = 'ETS-EC2-instances-MF-AWS-CICD'
  var command = "& C:\\BankDemo\\Test\\BankDemo-Tests.bat"
  var instanceId
  var codepipeline = new AWS.CodePipeline()
  var runEc2CommandOneSuccess = false

  console.log('Received event ', JSON.stringify(event))
  try {
    jobId = event['CodePipeline.job'].id
    console.log('Found CodePipeline job with ID: ', jobId)
    jobUserParameters = JSON.parse(event['CodePipeline.job'].data.actionConfiguration.configuration.UserParameters)
    console.log('Found CodePipeline job with parameters: ', jobUserParameters)
    instanceTagKey = jobUserParameters.Ec2TagKey
    instanceTagValue = jobUserParameters.Ec2TagValue
    command = jobUserParameters.Ec2Command
    if ((!instanceTagKey) || (!instanceTagValue) || (!command)) { console.error('Error trying to retrieve CodePipeline user parameters. In CodePipeline, the User Parameters must be in JSON format following {"Ec2TagKey": "myTagKey", "Ec2TagValue": "myTagValue", "Ec2Command": "& C:\\myPath\\myCommand.bat"}.' ) }
  } catch (error) {
    console.error('Error trying to retrieve CodePipeline parameters. In CodePipeline, the User Parameters must be in JSON format following {"Ec2TagKey": "myTagKey", "Ec2TagValue": "myTagValue", "Ec2Command": "& C:\\myPath\\myCommand.bat"}. Continuing with default Lambda function values. Caught error: ', error.toString())
  }
  try {

    var tagFilter = { Filters: [ { Name: 'tag:' + instanceTagKey, Values: [ instanceTagValue ] } ] };
    const instancesData = await ec2.describeInstances(tagFilter).promise();
    instancesData.Reservations.forEach(reservation => {
      reservation.Instances.forEach(instance => {
        //console.log('Looking at instance: ', instance.InstanceId)
        if (instance.State.Code === 16) {
          // 0: pending, 16: running, 32: shutting-down, 48: terminated, 64: stopping, 80: stopped
          instanceIds.push(instance.InstanceId);
          console.log('Instance found running with tag { ' + instanceTagKey + ': ' + instanceTagValue + ' } : ', instance.InstanceId)
        }
      })
    })
  }

}
```

```

    });
    //console.log('instanceIds: ', instanceIds)
    if (instanceIds.length == 0) {
        console.error('No instance found with status Running and tag { ', instanceTagKey, ': ', instanceTagValue, ' }')
    } else {
        for (instanceId of instanceIds) {
            // Send command to EC2 instance via SSM
            const sendCommandPromise = ssm.sendCommand({
                DocumentName: "AWS-RunPowerShellScript",
                InstanceIds: [ instanceId ],
                Parameters: { "commands": [ command ], "workingDirectory": [ "" ] },
                TimeoutSeconds: timeoutSSM
            }).promise();
            console.log(instanceId, ' - SSM command sent to instance')
            //console.log(instanceId, ' - PowerShell command sent: ', command)
            const sendCommandResult = await sendCommandPromise
            const commandId = sendCommandResult.Command.CommandId

            var commandStatus = ''
            var getCommandInvocationResult
            do {
                console.log(instanceId, ' - Waiting for SSM response...')
                await new Promise(resolve => setTimeout(resolve, waitInterval * 1000));
                const getCommandInvocationPromise = ssm.getCommandInvocation({ CommandId: commandId, InstanceId: instanceId }).promise();
                getCommandInvocationResult = await getCommandInvocationPromise
                //console.log('getCommandInvocationResult: ', getCommandInvocationResult)
                commandStatus = getCommandInvocationResult.Status
                console.log(instanceId, ' - SSM command status: ', commandStatus)
                //if (commandStatus == 'Success') { console.log('getCommandInvocationResult: ', getCommandInvocationResult) }
                //if (commandStatus != 'Success') { console.log('getCommandInvocationResult: ', getCommandInvocationResult) }
                // while ((commandStatus != 'Success') && (commandStatus != 'Cancelled') && (commandStatus != 'TimedOut') && (commandStatus != 'Failed'));
                if (commandStatus == 'Success') {
                    if (getCommandInvocationResult.StandardErrorContent.length == 0) {
                        runEc2CommandOneSuccess = true
                        console.log(instanceId, ' - Command successfully executed via SSM.')
                        console.log(instanceId, ' - Command StdOut: ', getCommandInvocationResult.StandardOutputContent)
                        console.log(instanceId, ' - Command StdErr: ', getCommandInvocationResult.StandardErrorContent)
                    } else {
                        console.error(instanceId, ' - Command executed via SSM, but generated an error.')
                        console.error(instanceId, ' - Command StdOut: ', getCommandInvocationResult.StandardOutputContent)
                        console.error(instanceId, ' - Command StdErr: ', getCommandInvocationResult.StandardErrorContent)
                    }
                } else {
                    console.error(instanceId, ' - SSM command failed.')
                    console.error(instanceId, ' - SSM ResponseCode: ', getCommandInvocationResult.ResponseCode)
                    console.error(instanceId, ' - SSM Status: ', getCommandInvocationResult.Status)
                    console.error(instanceId, ' - SSM StatusDetails: ', getCommandInvocationResult.StatusDetails)
                }
            }
        }

        if (runEc2CommandOneSuccess) {
            console.log('One command execution on an EC2 instance was successful.')
            if (jobId) {
                console.log('Sending putJobSuccessResult to CodePipeline.')
                await codepipeline.putJobSuccessResult({ jobId }).promise()
            }
        } else {
            console.error('Command execution on EC2 instance(s) was unsuccessful.')
            if (jobId) {
                console.log('Sending putJobFailureResult to CodePipeline.')
                await codepipeline.putJobFailureResult({ jobId, failureDetails: { message: 'Script error. See Command StdErr for details', type: 'JobFailed', externalExecutionId: context.invokeId } }).promise()
            }
        }
    }
} catch (error) {
    console.error('Error caught during Lambda function execution:', error.toString())
    if (jobId) {
        console.log('Sending putJobFailureResult to CodePipeline.')
        await codepipeline.putJobFailureResult({ jobId, failureDetails: { message: error.toString(), type: 'JobFailed', externalExecutionId: context.invokeId } }).promise()
    } else {
        throw error
    }
}
}

```

Tailor the instanceTagKey, instanceTagValue, command variables to your specific environment.

For the Execution role, verify the existing role Lambda-SSM-EC2-Logs is selected.

Edit the Basic setting timeout, select 5 min 30 sec.

Save the function.

Click Test to run this new function. Verify it runs successfully.

## 6.4 TEST – AWS CodePipeline configuration

In CodePipeline, go to the MF-AWS-CICD-Pipeline and click Edit.

Add a stage after the Deploy stage.

Add stage

Stage name

Test

No more than 100 characters

Cancel Add stage

For this new Test stage, click Add action group.

Action name is: Test

Action provider is: AWS Lambda

Region is the region used for the ETS EC2 instance.

Function name is: CodePipeline-Lambda-SSM-EC2-RunPowerShellScript

Under User parameters, enter the JSON string specifying the tag and the test command:

```
{ "Ec2TagKey": "CodeDeployGroup", "Ec2TagValue": "ETS-EC2-instances-MF-AWS-CICD",  
"Ec2Command": "& C:\\BankDemo\\Test\\BankDemo-Tests.bat"}
```

Edit action

Action name

Choose a name for your action

Test

No more than 100 characters

Action provider

AWS Lambda

Region

US East - (N. Virginia)

Input artifacts

Choose an input artifact for this action. [Learn more](#)

Add

No more than 100 characters

Function name

Choose a function that you have already created in the AWS Lambda console. Or create a function in the Amazon Lambda console and then return to this task.

Q. CodePipeline-Lambda-SSM-EC2-RunPowerShellScript X C

User parameters - optional

This string will be used in the event data parameter passed to the handler in AWS Lambda.

{"Ec2TagKey": "CodeDeployGroup", "Ec2TagValue": "ETS-EC2-instances-MF-AWS-CICD", "Ec2Command": "& C:\\BankDemo\\Test\\BankDemo-Tests.bat"}

Variable namespace - optional

Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. [Learn more](#)

Add

No more than 100 characters

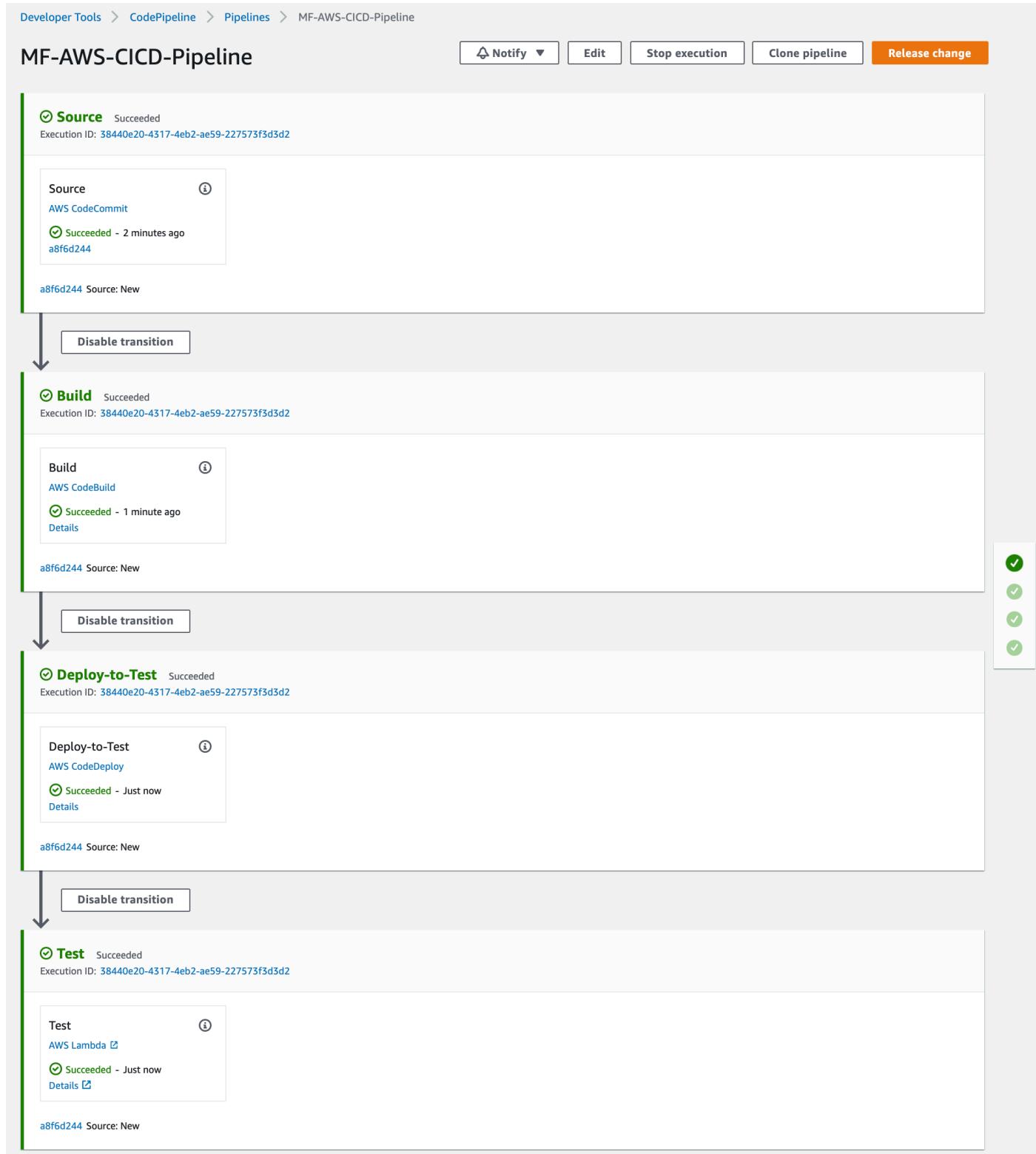
Cancel Done

Click Done.

Click Done for the Stage.

Click Save for the pipeline.

Click Release change to re-run the last code change through the pipeline with the new Test stage.



# 7 DEPLOY to PROD stage

---

## 7.1 DEPLOY to PROD – AWS Lambda configuration

We now create a Lambda function to pull the changed files from CodeCommit and process them for deployment to production.

In IAM, create an IAM role named Lambda-CodeCommit-CodePipeline-Logs which contains an inline policy with this JSON:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "codecommit:Get*",  
                "codecommit>List*",  
                "codecommit:DescribePullRequestEvents",  
                "codecommit:GitPull",  
                "codecommit:BatchGetRepositories",  
                "codecommit:BatchGetPullRequests",  
                "codepipeline:PutJobSuccessResult",  
                "codepipeline:PutJobFailureResult",  
                "logs>CreateLogGroup",  
                "logs>CreateLogStream",  
                "logs:PutLogEvents"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

## Summary

[Delete role](#)

<b>Role ARN</b>	arn:aws:iam::954254376221:role/Lambda-CodeCommit-CodePipeline-Logs <a href="#">Edit</a>
<b>Role description</b>	Allows Lambda functions to call AWS services on your behalf.   <a href="#">Edit</a>
<b>Instance Profile ARNs</b>	<a href="#">Edit</a>
<b>Path</b>	/
<b>Creation time</b>	2020-02-24 20:48 EST
<b>Last activity</b>	Not accessed in the tracking period
<b>Maximum CLI/API session duration</b>	1 hour <a href="#">Edit</a>

[Permissions](#) [Trust relationships](#) [Tags](#) [Access Advisor](#) [Revoke sessions](#)
▼ Permissions policies (1 policy applied)
[Attach policies](#)[+ Add inline policy](#)

Policy name	Policy type	X
Lambda-CodeCommit-CodePipeline-Logs-policy	Inline policy	X
<div style="border: 1px solid #ccc; padding: 10px;"> <div style="display: flex; justify-content: space-between;"> <span>Policy summary</span> <span>{ JSON }</span> <span>Edit policy</span> </div> <pre> 5   "Effect": "Allow", 6   "Action": [ 7     "codecommit:Get*", 8     "codecommit&gt;List*", 9     "codecommitDescribePullRequestEvents", 10    "codecommitGitPull", 11    "codecommitBatchGetRepositories", 12    "codecommitBatchGetPullRequests", 13    "codepipelinePutJobSuccessResult", 14    "codepipelinePutJobFailureResult", 15    "logsCreateLogGroup", 16    "logsCreateLogStream", 17    "logsPutLogEvents" 18  ], 19  "Resource": "*" </pre> <div style="float: right;"> <a href="#">Simulate policy</a> </div> </div>		

In Lambda, create a Lambda function named CodePipeline-Lambda-CodeCommit-DeployToProd.

Select Runtime Node.js 12.x.

Select the Lambda-CodeCommit-CodePipeline-Logs existing role.

**Basic information**

Function name  
Enter a name that describes the purpose of your function.

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info  
Choose the language to use to write your function.

Permissions Info  
Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

▼ Choose or create an execution role

Execution role  
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Create a new role with basic Lambda permissions  
 Use an existing role  
 Create a new role from AWS policy templates

Existing role  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

[View the Lambda-CodeCommit-CodePipeline-Logs role on the IAM console.](#) [C](#)

[Cancel](#) [Create function](#)

Click Create function.

Enter code inline index.js:

```
const AWS = require('aws-sdk');
var jobId
var commitId = '4aae84ccb0c0d59d9aa133dc85f076357993822c'
var repositoryName = 'MF-AWS-CICD-SCM'
var jobUserParameters
var differencesData
var difference

exports.handler = async (event, context) => {
    var codepipeline = new AWS.CodePipeline()
    var codecommit = new AWS.CodeCommit()

    console.log('Received event ', JSON.stringify(event))
    try {
        jobId = event["CodePipeline.job"].id
        console.log('Found CodePipeline job with ID: ', jobId)
        jobUserParameters = JSON.parse(event["CodePipeline.job"].data.actionConfiguration.configuration.UserParameters)
        console.log('Found CodePipeline job with parameters: ', jobUserParameters)
        commitId = jobUserParameters.commitId
        console.log('Found CodeCommit CommitId: ', commitId)
        repositoryName = jobUserParameters.repositoryName
        console.log('Found CodeCommit RepositoryName: ', repositoryName)
        if ((!commitId) || (!repositoryName)) { console.error('Error trying to retrieve CodePipeline user parameters. In CodePipeline, the User Parameters must be { "commitId": "#${SourceVariables.CommitId}" , "repositoryName": "#${SourceVariables.RepositoryName}" }.' ) }
        } catch (error) {
            console.error('Error trying to retrieve CodePipeline parameters. In CodePipeline, the User Parameters must be { "commitId": "#${SourceVariables.CommitId}" , "repositoryName": "#${SourceVariables.RepositoryName}" }. Continuing with default Lambda function values. Caught error: ', error.toString())
        }

        try {
            console.log('commitId:', commitId)
            const commitData = await codecommit.getCommit({ commitId: commitId, repositoryName: repositoryName }).promise()
            //console.log(commitData)
            console.log('Commit message:', commitData.commit.message)
            if (commitData.commit.parents[0].length > 0) {
                const priorCommitId = commitData.commit.parents[0]
                console.log('priorCommitId:', priorCommitId)
                differencesData = await codecommit.getDifferences({ repositoryName: repositoryName, afterCommitSpecifier: commitId, beforeCommitSpecifier: priorCommitId }).promise()
            } else {
                differencesData = await codecommit.getDifferences({ repositoryName: repositoryName, afterCommitSpecifier: commitId })
            }
            //console.log('differencesData:', differencesData)

            for (difference of differencesData.differences) {
                //console.log('Processing difference:', difference)
                if (difference.changeType == 'A') {
                    console.log('Processing DeployToProd file add:', difference.afterBlob.path )
                }
            }
        }
    }
}
```

```

        console.log('Processing DeployToProd file add with blobId:', difference.afterBlob.blobId )
        const blobData = await codecommit.getBlob({ blobId: difference.afterBlob.blobId, repositoryName: repositoryName
    }).promise()
        const blobContent = blobData.content
        console.log('blobContent:', blobContent)
        // Add your code addition logic here

    } else if (difference.changeType == 'M') {
        console.log('Processing DeployToProd file modify:', difference.afterBlob.path )
        console.log('Processing DeployToProd file modify with blobId:', difference.afterBlob.blobId )
        const blobData = await codecommit.getBlob({ blobId: difference.afterBlob.blobId, repositoryName: repositoryName
    }).promise()
        const blobContent = blobData.content
        console.log('blobContent:', blobContent)
        // Add your code modification logic here

    } else if (difference.changeType == 'D') {
        console.log('Processing DeployToProd file delete:', difference.afterBlob.path )
        console.log('Processing DeployToProd file delete with blobId:', difference.afterBlob.blobId )
        // Add your code deletion logic here

    } else {
        console.log('changeType not processed: ', difference.changeType)
    }
}

if (jobId) {
    console.log('Sending putJobSuccessResult to CodePipeline.')
    await codepipeline.putJobSuccessResult({ jobId }).promise()
}

} catch (error) {
    console.error('Error caught during Lambda function execution:', error.toString())
    if (jobId) {
        console.log('Sending putJobFailureResult to CodePipeline.')
        await codepipeline.putJobFailureResult({ jobId, failureDetails: { message: error.toString(), type: 'JobFailed', externalExecutionId: context.invokedId }}).promise()
    } else {
        throw error
    }
}
}

```

Edit the Basic setting timeout, select 1min.

Save the function.

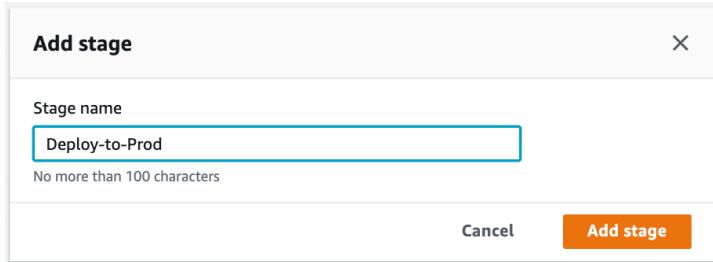
Click Test to run this new function. Verify it runs successfully

## 7.2 DEPLOY to PROD – AWS CodePipeline configuration

In CodePipeline, we're now going to add the new stage to deploy the code to the production environment.  
In CodePipeline, then Pipelines, then Pipeline, select MF-AWS-CICD-Pipeline.

Click Edit.

After the last stage, at the bottom, click Add stage.



Click Add action group.

The Action name is: Deploy-to-Prod

The Action provider is: AWS Lambda

Select the proper region for your environment.

Input artifacts is SourceArtifact

Function name is: CodePipeline-Lambda-CodeCommit-DeployToProd

User parameters is: { "commitId": "\${SourceVariables.CommitId}", "repositoryName": "\${SourceVariables.RepositoryName}" }

**Edit action**

Action name  
Choose a name for your action  
  
No more than 100 characters

Action provider

Region

Input artifacts  
Choose an input artifact for this action. [Learn more](#)

No more than 100 characters

Function name  
Choose a function that you have already created in the AWS Lambda console. Or create a function in the Amazon Lambda console and then return to this task.

User parameters - optional  
This string will be used in the event data parameter passed to the handler in AWS Lambda.

Variable namespace - optional  
Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. [Learn more](#)

Output artifacts  
Choose a name for the output of this action.

No more than 100 characters

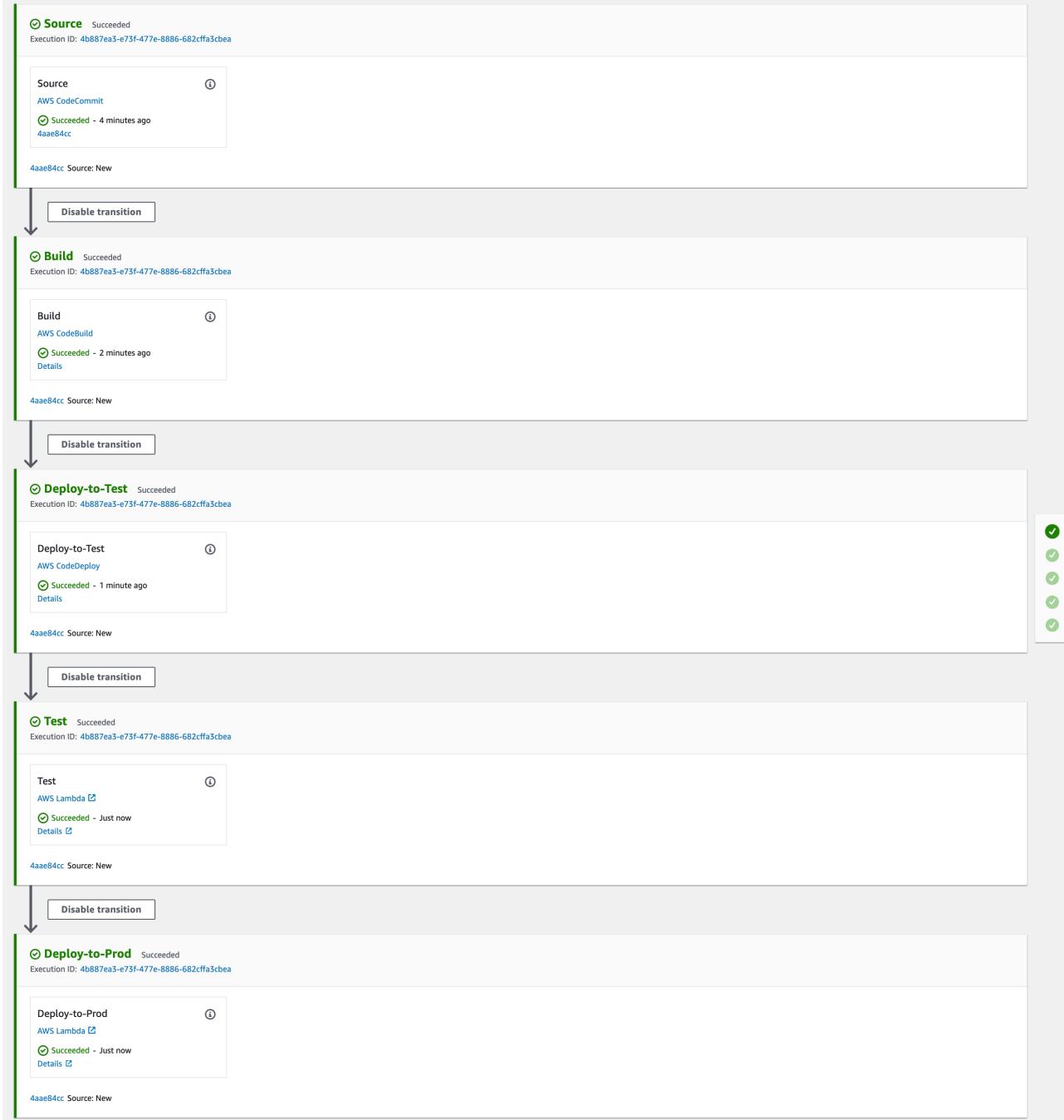
Click Done.

Click Done to save the new stage.

Click Save to save the changed pipeline.

In order to test, you can either commit and push to upstream a new change from EDz, or click Release change on the pipeline itself.

## MF-AWS-CICD-Pipeline

[Notify](#) [Edit](#) [Stop execution](#) [Clone pipeline](#) [Release change](#)


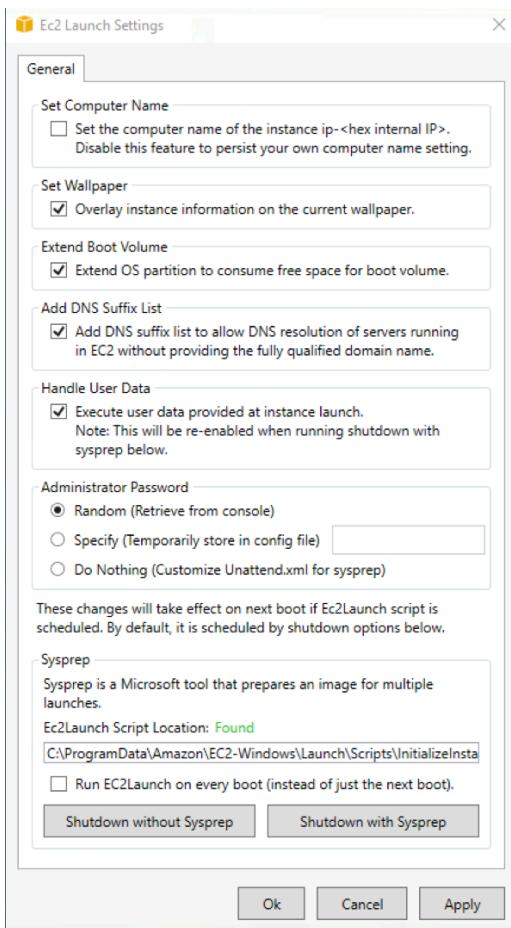
# 8 Appendix

## 8.1 Configure instance for new Administrator random password

C:\ProgramData\Amazon\EC2-Windows\Launch\Config\LaunchConfig.json

```
{  
    "SetComputerName": false,  
    "SetMonitorAlwaysOn": true,  
    "SetWallpaper": true,  
    "AddDnsSuffixList": true,  
    "ExtendBootVolumeSize": true,  
    "HandleUserData": true,  
    "AdminPasswordType": "Random",  
    "AdminPassword": ""  
}
```

C:\ProgramData\Amazon\EC2-Windows\Launch\Settings\Ec2LaunchSettings.exe



Shutdown with Sysprep then create new AMI from stopped instance.

## 8.2 Powershell script to compile Bankdemo on EDz

Bankdemo-build.ps1

```

### Documentation: https://www.microfocus.com/documentation/enterprise-developer/ed50pu2/ED-Eclipse/HRCMRHCOML0L.html

### Variables
$env:SOURCE_COBOL_FOLDER = "C:\Users\Administrator\TestBuild\Bankdemo\Sources\cbl"
$env:COBOL_EXE_PATH_FILE = "C:\Program Files (x86)\Micro Focus\Enterprise Developer\bin64\cobol.exe"
$env:CBLLINK_EXE_PATH_FILE = "C:\Program Files (x86)\Micro Focus\Enterprise Developer\bin64\cbllink.exe"
$env:COBOL_COMPILER_DIRECTIVES = "preprocess(EXCI),CICSCECM()"
$env:COBDIR="C:\Program Files (x86)\Micro Focus\Enterprise Developer\;$env:COBDIR" # where to find the message file
$env:COBCPY = "C:\Program Files (x86)\Micro Focus\Enterprise Developer\cpylib;$env:COBCPY" # where to find copybooks such as DFHAID

echo "=====
echo "Environment Variables:"
echo "SOURCE_COBOL_FOLDER: $env:SOURCE_COBOL_FOLDER"
echo "COBOL_COMPILER_DIRECTIVES: $env:COBOL_COMPILER_DIRECTIVES"
echo "COBOL_EXE_PATH_FILE: $env:COBOL_EXE_PATH_FILE"
echo "CBLLINK_EXE_PATH_FILE: $env:CBLLINK_EXE_PATH_FILE"
echo "COBDIR: $env:COBDIR"
echo "COBCPY: $env:COBCPY"
echo ""

echo "=====
echo "Compiling begins"
foreach ($f in Get-ChildItem $env:SOURCE_COBOL_FOLDER -Filter "*.cbl") {
    $CBL_FILE = $f.FullName
    echo -----
    echo "Compiling $CBL_FILE..."
    & $env:COBOL_EXE_PATH_FILE "$CBL_FILE,,,,$env:COBOL_COMPILER_DIRECTIVES"

}
echo -----
echo "Compiling ends"
echo ""

echo "=====
echo "Linking begins"

foreach ($f in Get-ChildItem "." -Filter "*.obj") {
    $OBJ_FILE = $f.FullName
    echo -----
    echo "Linking $OBJ_FILE..."
    & $env:CBLLINK_EXE_PATH_FILE -d $OBJ_FILE

}
echo -----
echo "Linking ends"

```