Morris & Opazo

# Simplify refactoring of .NET applications

aws PARTNER Advanced Tier Services

- Public Sector
- Immersion Day
- Solution Provider
- AWS Lambda Delivery
- Amazon Kinesis Delivery
- Amazon Redshift Delivery
- Amazon API Gateway Delivery
- Data & Analytics Services Competency
- Amazon EC2 for Windows Server Delivery

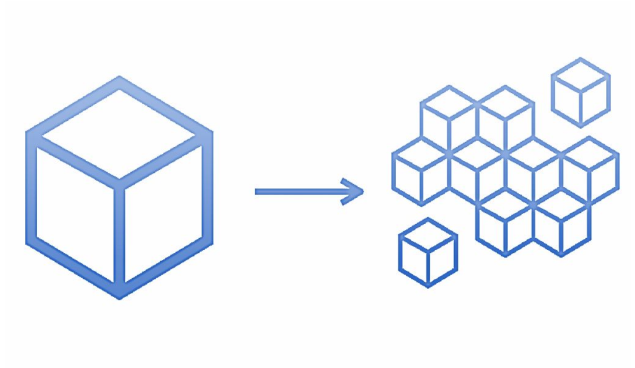
morrisopazo.com
contacto@morrisopazo.com

# AWS MICROSERVICE EXTRACTOR FOR.NET

Morris & Opazo

## What is AWS Microservice Extractor for.Net?

It is an Amazon assistance tool that seeks to simplify the process of modernizing or modifying code in a .Net application without changing its external behavior, but only its internal structure, leaving it divided into smaller and more independent codes, through a decoupling. This allows you to progressively innovate in new technologies.
AWS Microservice Extractor for .NET supports .NET Framework and .NET Core ASP.NET web services applications



## ¿What does it mean for an App to be .Net?

.Net is a platform created by Microsoft that allows the creation and execution of applications using a series of languages, implementations, tools and libraries for its development.

.

# ANALYZING THE DEPENDENCIES

## What does the AWS microservice Extractor for. Net?

This tool for .Net applications analyzes the source code and provides a visual analysis through a graph of the execution metrics with their dependencies, that is, it manages to visualize the performance of the software and the dependencies of services or libraries required from other programs for good App execution.

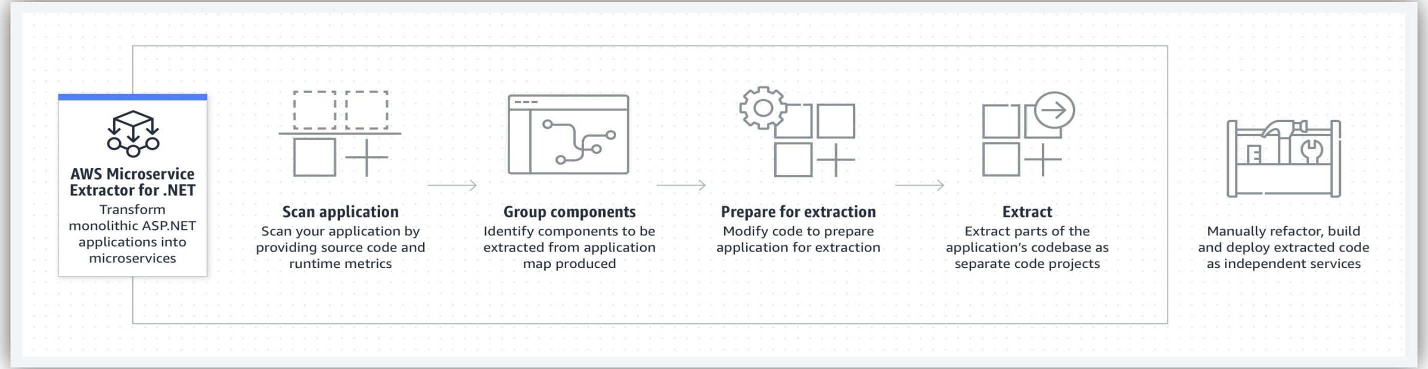
Having an overview of the application and its service calls will help you make informed decisions about the structure, it helps you understand the principles of Domain-Driven Design, which helps identify patterns in an application between its components and its calls. . Essential to avoid failure in the transformation of the code structure.
Aws microservice extractor for.net also allows us to label and group these dependencies to make way for the extraction.

# PREPARING THE EXTRACTION

Morris & Opazo

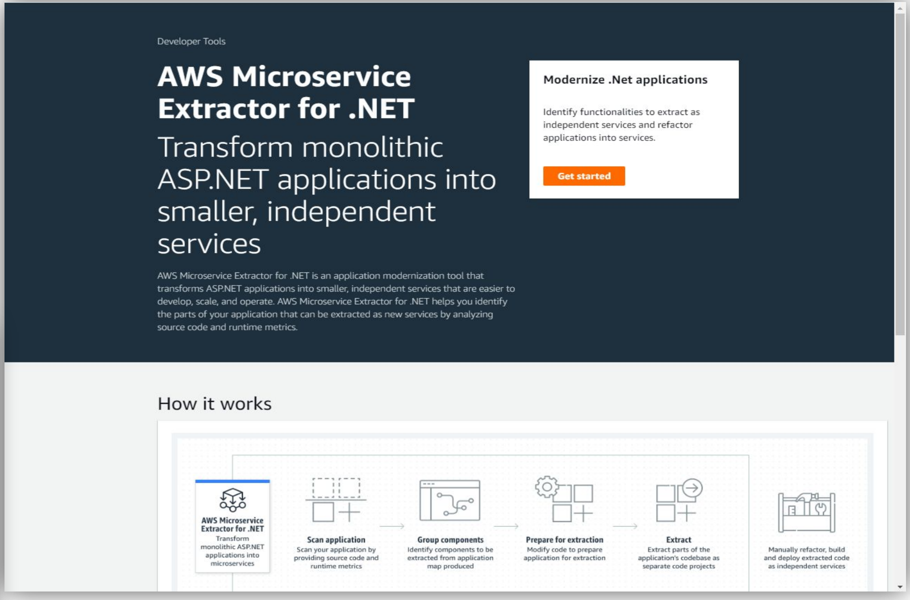
Microservice Extractor for.Net Provides a guide to help break down the base code, preparing it for extraction into smaller services, recognized as 'Islands' in the application visualizations below.
Once the scanner has been carried out, the refactoring of the code and the visible dependencies, it is possible to manually eliminate each one of them, to prepare parts of the application and carry out the extraction.
The source code would be in units that teams can develop, build, deploy, and operate as stand-alone services with their choice of tools.

## How does it operate?


AWS Microservice Extractor for .NET
Transform monolithic ASP.NET applications into microservices
Scan application
Scan your application by providing source code and runtime metrics
Group components
Identify components to be extracted from application map produced
Prepare for extraction
Modify code to prepare application for extraction
Extract
Extract parts of the application's codebase as separate code projects
Manually refactor, build and deploy extracted code as independent services

# HOW TO IMPLEMENT MICROSERVICE EXTRACTOR?

Once the service is downloaded, you will see the main page



Developer Tools

**AWS Microservice Extractor for .NET**

Transform monolithic ASP.NET applications into smaller, independent services

AWS Microservice Extractor for .NET is an application modernization tool that transforms ASP.NET applications into smaller, independent services that are easier to develop, scale, and operate. AWS Microservice Extractor for .NET helps you identify the parts of your application that can be extracted as new services by analyzing source code and runtime metrics.

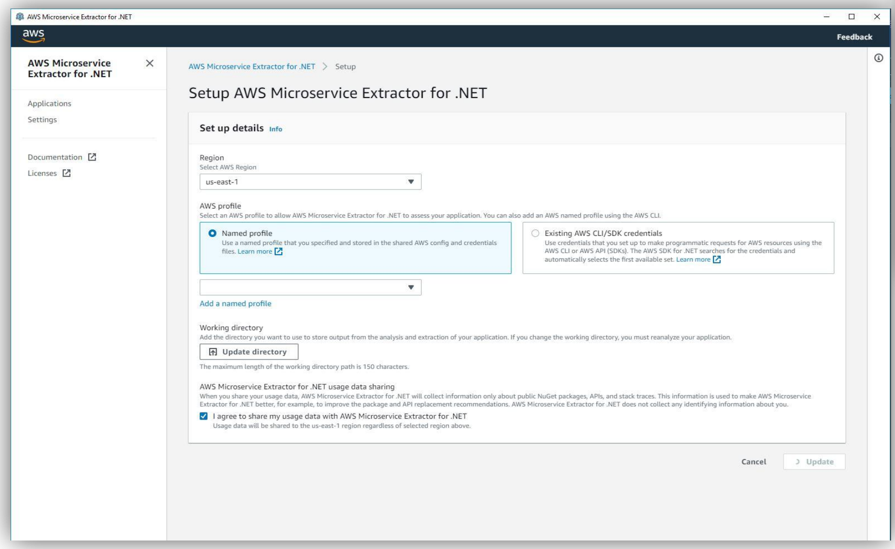
**Modernize .Net applications**

Identify functionalities to extract as independent services and refactor applications into services.

Get started

How it works

AWS Microservice Extractor for .NET
Transform monolithic ASP.NET applications into microservices
Scan application
Scan your application by providing source code and runtime metrics
Group components
Identify components to be extracted from application map produced
Prepare for extraction
Modify code to prepare application for extraction
Extract
Extract parts of the application's codebase as separate code projects
Manually refactor, build and deploy extracted code as independent services



AWS Microservice Extractor for .NET

Feedback

AWS Microservice Extractor for .NET

- Applications
- Settings
- Documentation
- Licenses

AWS Microservice Extractor for .NET > Setup

Setup AWS Microservice Extractor for .NET

**Set up details** Info

Region
Select AWS Region
us-east-1

AWS profile
Select an AWS profile to allow AWS Microservice Extractor for .NET to assess your application. You can also add an AWS named profile using the AWS CLI.

● Named profile
Use a named profile that you specified and stored in the shared AWS config and credentials files. Learn more

○ Existing AWS CLI/SDK credentials
Use credentials that you set up to make programmatic requests for AWS resources using the AWS CLI or AWS API (SDKs). The AWS SDK for .NET searches for the credentials and automatically selects the first available set. Learn more

Add a named profile

Working directory
Add the directory you want to use to store output from the analysis and extraction of your application. If you change the working directory, you must reanalyze your application.

Update directory

The maximum length of the working directory path is 150 characters.

AWS Microservice Extractor for .NET usage data sharing
When you share your usage data, AWS Microservice Extractor for .NET will collect information only about public NuGet packages, APIs, and stack traces. This information is used to make AWS Microservice Extractor for .NET better, for example, to improve the package and API replacement recommendations. AWS Microservice Extractor for .NET does not collect any identifying information about you.

☑ I agree to share my usage data with AWS Microservice Extractor for .NET
Usage data will be shared to the us-east-1 region regardless of selected region above.

Cancel | Update

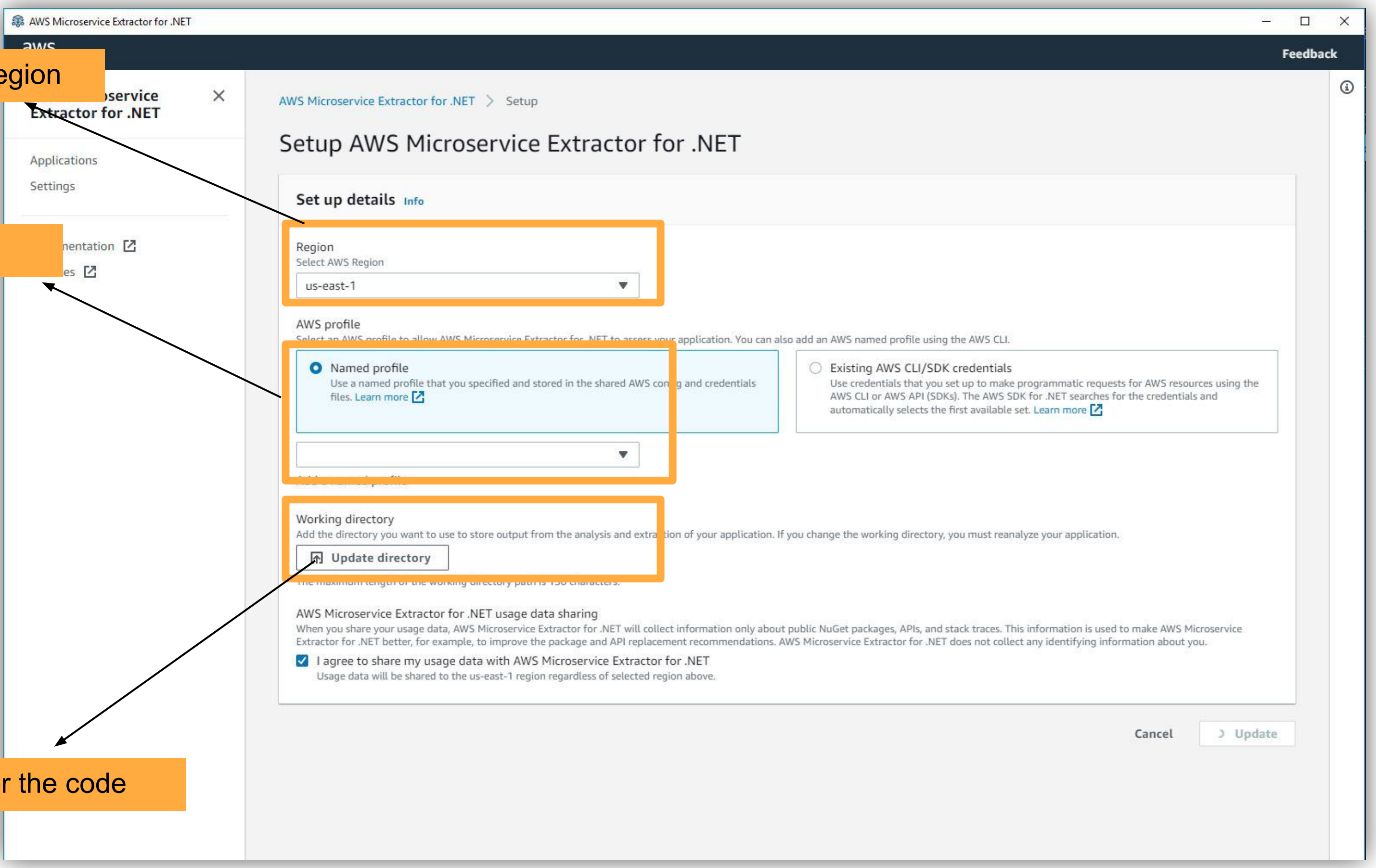
And after starting is the setUp to configure

# INITIAL SET-UP

**In the initial setup configuration is:**

- Region to which you will be connected with AWS.
- An AWS profile for API calls (user identity) or you can create a keyed profile from IAM.
- Working directory where the working copy of your code will be stored. It is recommended that the directory name be short to avoid difficulties with the compilation since the microservice extractor for.net works with SMBUIL to compile and has some limitations.
- Msbuild is one of the prerequisites for using the Microservice extractor because the code is built at runtime (if your computer has Visual Studio installed the path will be filled in automatically).
- Finally, you have the option to share data with AWS, in order to improve the service, if not, you can uncheck it.

# SETTING



AWS Microservice Extractor for .NET

Feedback

Extractor for .NET

Applications

Settings

AWS Microservice Extractor for .NET > Setup

## Setup AWS Microservice Extractor for .NET

**Set up details** Info

Region
Select AWS Region
us-east-1

AWS profile
Select an AWS profile to allow AWS Microservice Extractor for .NET to access your application. You can also add an AWS named profile using the AWS CLI.

● Named profile
Use a named profile that you specified and stored in the shared AWS config and credentials files. Learn more

○ Existing AWS CLI/SDK credentials
Use credentials that you set up to make programmatic requests for AWS resources using the AWS CLI or AWS API (SDKs). The AWS SDK for .NET searches for the credentials and automatically selects the first available set. Learn more

Working directory
Add the directory you want to use to store output from the analysis and extraction of your application. If you change the working directory, you must reanalyze your application.

Update directory

The maximum length of the working directory path is 150 characters.

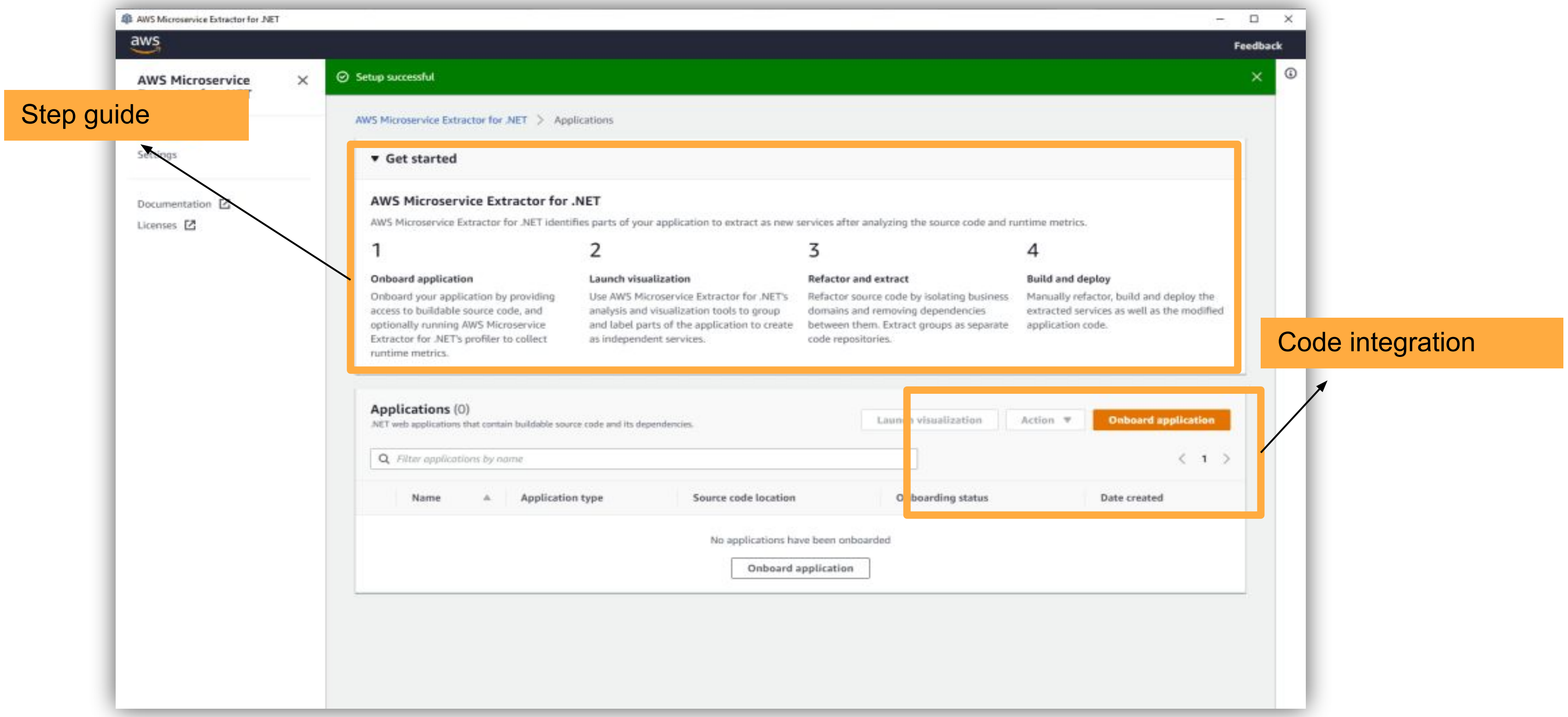
AWS Microservice Extractor for .NET usage data sharing
When you share your usage data, AWS Microservice Extractor for .NET will collect information only about public NuGet packages, APIs, and stack traces. This information is used to make AWS Microservice Extractor for .NET better, for example, to improve the package and API replacement recommendations. AWS Microservice Extractor for .NET does not collect any identifying information about you.

☑ I agree to share my usage data with AWS Microservice Extractor for .NET
Usage data will be shared to the us-east-1 region regardless of selected region above.

Cancel    Update

# STEP BY STEP

You will be able to see the steps to follow and the option to incorporate your code



Get started

AWS Microservice Extractor for .NET

AWS Microservice Extractor for .NET identifies parts of your application to extract as new services after analyzing the source code and runtime metrics.

1

**Onboard application**

Onboard your application by providing access to buildable source code, and optionally running AWS Microservice Extractor for .NET's profiler to collect runtime metrics.

2

**Launch visualization**

Use AWS Microservice Extractor for .NET's analysis and visualization tools to group and label parts of the application to create as independent services.

3

**Refactor and extract**

Refactor source code by isolating business domains and removing dependencies between them. Extract groups as separate code repositories.

4

**Build and deploy**

Manually refactor, build and deploy the extracted services as well as the modified application code.

Applications (0)

.NET web applications that contain buildable source code and its dependencies.

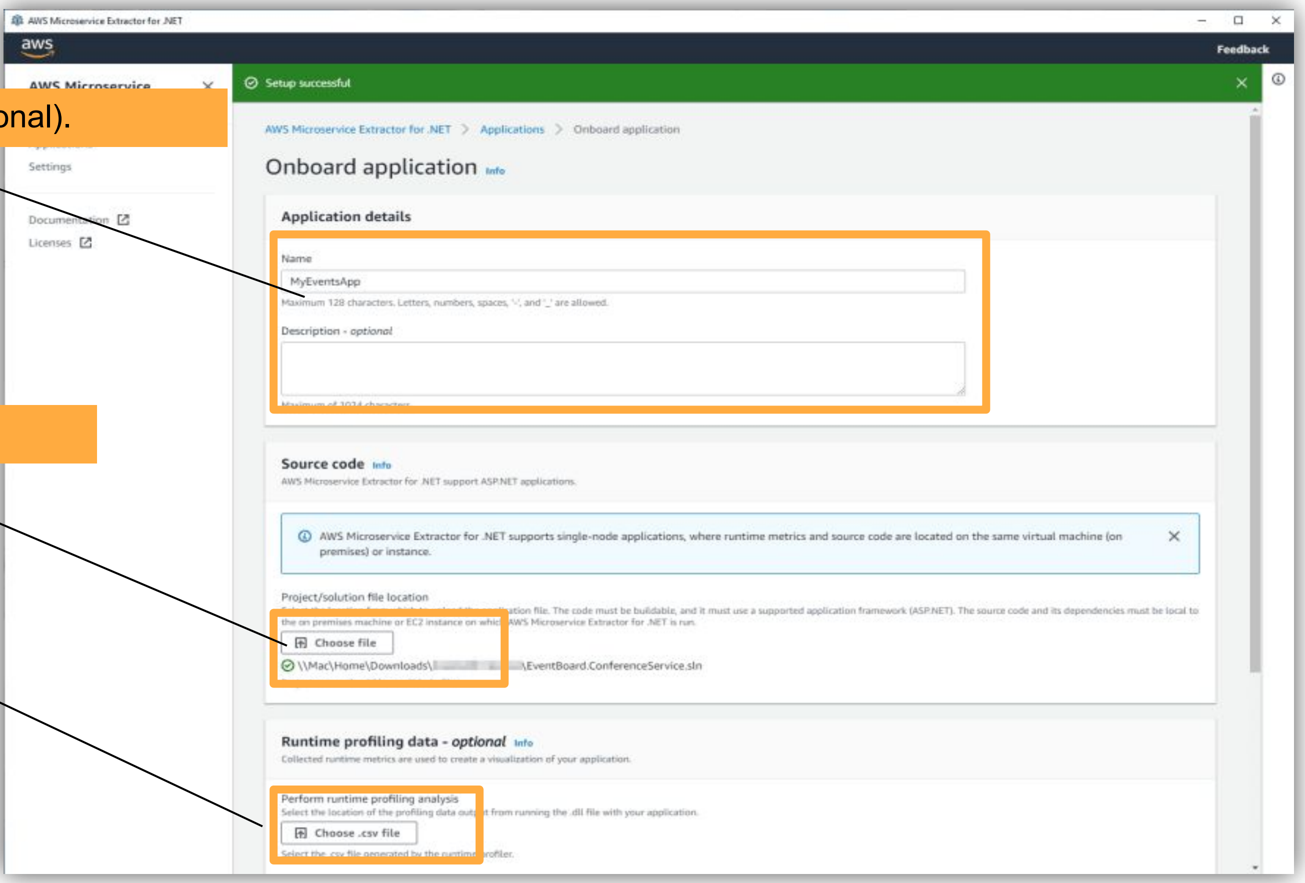
No applications have been onboarded

# CODE ON BOARD

App name and description (optional).

Solution file path.

This step is optional, but you can integrate runtime profiling to help you visualize and partition the components.


AWS Microservice Extractor for .NET

Feedback

AWS Microservice

Settings

Documentation

Licenses

Setup successful

AWS Microservice Extractor for .NET > Applications > Onboard application

## Onboard application Info

### Application details

Name

MyEventsApp

Maximum 128 characters. Letters, numbers, spaces, '-', and '_' are allowed.

Description - *optional*

Maximum of 1024 characters.

### Source code Info

AWS Microservice Extractor for .NET support ASP.NET applications.

AWS Microservice Extractor for .NET supports single-node applications, where runtime metrics and source code are located on the same virtual machine (on premises) or instance.

Project/solution file location

...ation file. The code must be buildable, and it must use a supported application framework (ASP.NET). The source code and its dependencies must be local to the on premises machine or EC2 instance on whic[h] AWS Microservice Extractor for .NET is run.

Choose file

\\Mac\Home\Downloads\[redacted]\EventBoard.ConferenceService.sln

### Runtime profiling data - *optional* Info

Collected runtime metrics are used to create a visualization of your application.

Perform runtime profiling analysis

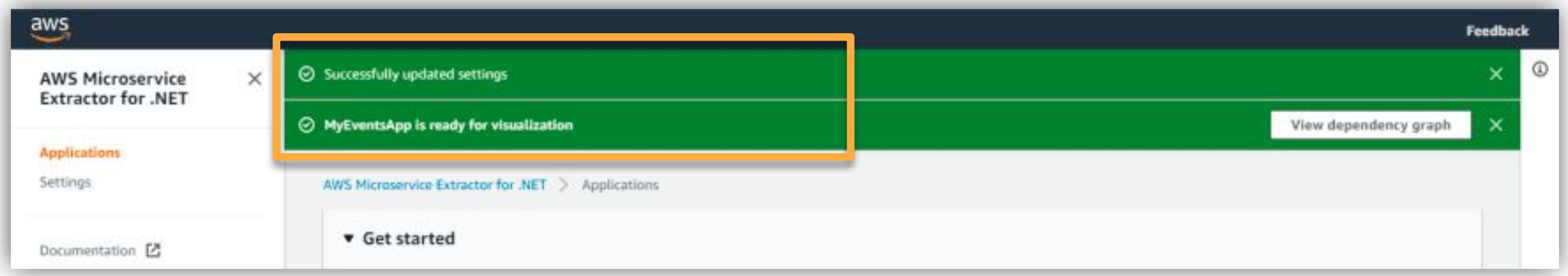
Select the location of the profiling data outp[u]t from running the .dll file with your application.

Choose .csv file

Select the .csv file generated by the runtim[e p]rofiler.

## DISPLAY

- When you have made the initial configurations, AWS microservice extractor for.net does a code scan assuming you have all the parts to not only analyze, but understand the logic as well.
- When the notification is green, it means that the code is ready to be visualized with its dependencies, classes and relationships

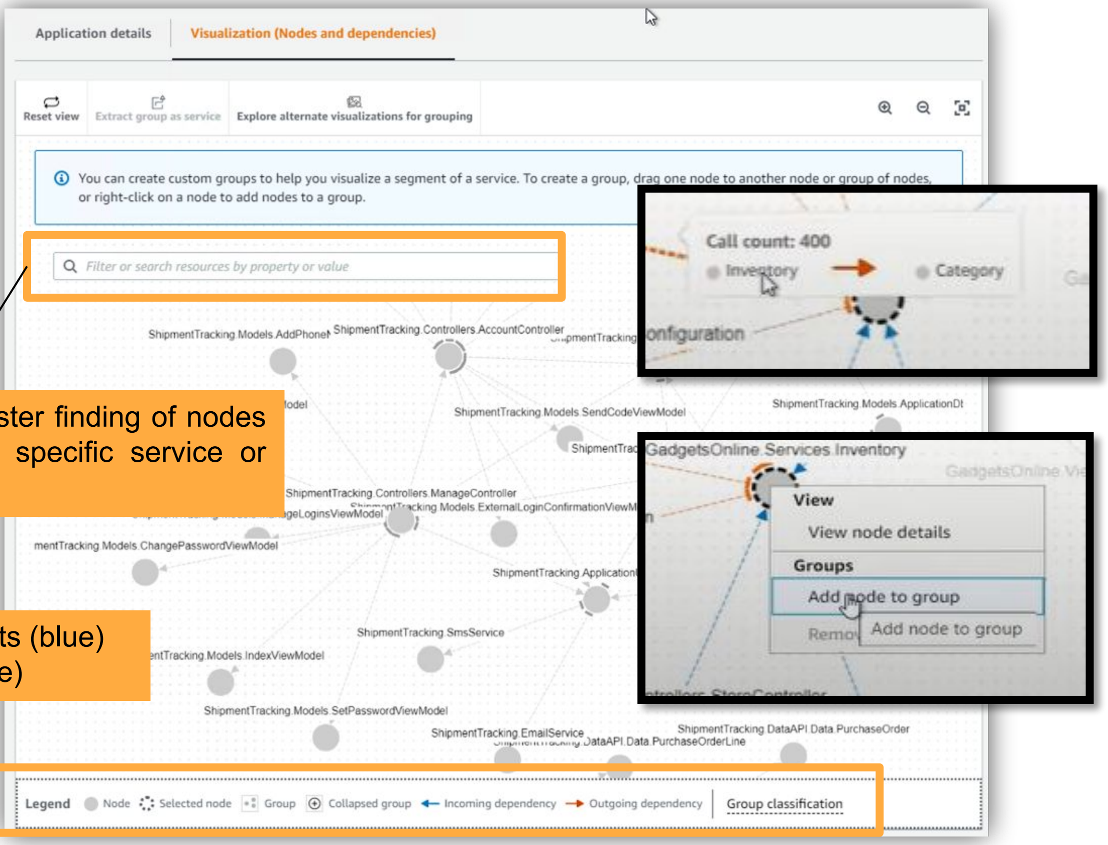

Successfully updated settings

MyEventsApp is ready for visualization

aws
Feedback
AWS Microservice Extractor for .NET
Applications
Settings
Documentation
View dependency graph
AWS Microservice Extractor for .NET > Applications
Get started

# DISPLAY GRAPH

The visualization is represented with this graph. Each circle is a component, the arrows are the incoming and outgoing relationships, in addition, you can see the call count of each component just by positioning the cursor over the node, in the same way with the right click you can group by domain.



Application details
Visualization (Nodes and dependencies)
Reset view
Extract group as service
Explore alternate visualizations for grouping
Filter or search resources by property or value
ShipmentTracking.Models.AddPhoneN
ShipmentTracking.Controllers.AccountController
ShipmentTracking.Models.SendCodeViewModel
ShipmentTracking.Models.ApplicationDt
ShipmentTracking.Controllers.ManageController
ShipmentTracking.Models.ExternalLoginConfirmationViewM
ManageLoginsViewModel
mentTracking.Models.ChangePasswordViewModel
ShipmentTracking.Application
ShipmentTracking.SmsService
entTracking.Models.IndexViewModel
ShipmentTracking.Models.SetPasswordViewModel
ShipmentTracking.EmailService
ShipmentTracking.DataAPI.Data.PurchaseOrderLine
ShipmentTracking.DataAPI.Data.PurchaseOrder
Call count: 400
Inventory
Category
onfiguration
GadgetsOnline.Services.Inventory
View
View node details
Groups
Add node to group
Remove
Add node to group
Legend
Node
Selected node
Group
Collapsed group
Incoming dependency
Outgoing dependency
Group classification

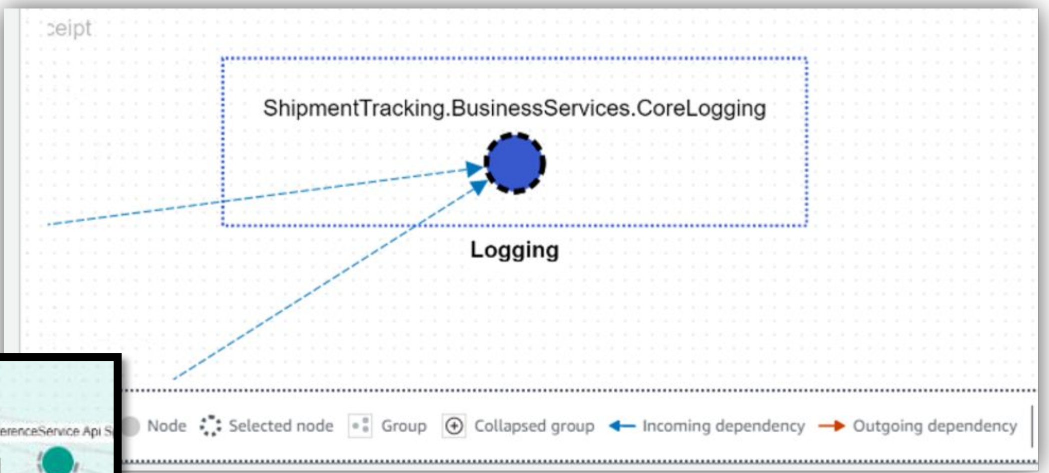
You can create custom groups to help you visualize a segment of a service. To create a group, drag one node to another node or group of nodes, or right-click on a node to add nodes to a group.

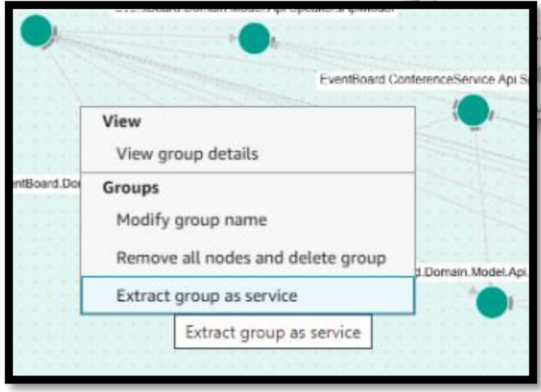
Integrated search bar for faster finding of nodes and classes to work on a specific service or group by islands

Sort by group, dependency inputs (blue) and dependency outputs (orange)

# EXTRACTION

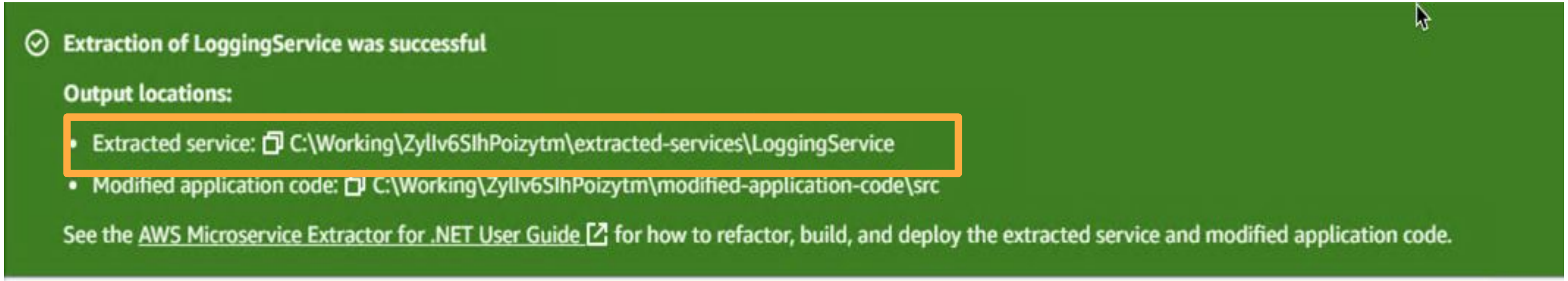
Once you have decided which part of the code is going to work, with the right click you can see the option to extract.





## EXTRACTION

If the extraction has been carried out successfully you will have a green notification at the top of the screen, with the path of the extracted microservices, which should be in a new file in the same directory as the main code.



Extraction of LoggingService was successful

Output locations:

- Extracted service: C:\Working\Zyllv6SlhPoizytm\extracted-services\LoggingService
- Modified application code: C:\Working\Zyllv6SlhPoizytm\modified-application-code\src

See the AWS Microservice Extractor for .NET User Guide for how to refactor, build, and deploy the extracted service and modified application code.

## BENEFITS OF MICROSERVICE EXTRACTOR

Faster identification of application parts to extract as services.

Create a dependency graph based on code analysis combination, eliminates the need to manually correlate the results of various tools to perform an analysis.

It partitions source code into units that teams can independently develop, build, deploy, and operate.

AWS Microservice Extractor for .NET allows you to tag and differentiate between your code and business processes by creating a domain design-based graph of your application.

## AWS RECOMMENDS

Aws recommends keeping in mind:

- Thoroughly review the application before moving it to production.
- Read the prerequisites and documentation carefully.
- Specify the source files of your applications to start a scan.
- Avoid long directory names to prevent compilation difficulties.
- Use the step guide and use the service in an assisted way.
- Download the free tool and see version compatibility.

Morris & Opazo

## Experts in Cloud Technology

aws
PARTNER
Advanced Tier
Services

- Public Sector
- Immersion Day
- Solution Provider
- AWS Lambda Delivery
- Amazon Kinesis Delivery

- Amazon Redshift Delivery
- Amazon API Gateway Delivery
- Data & Analytics Services Competency
- Amazon EC2 for Windows Server Delivery