Getting started with Julia on Amazon SageMaker

Step-by-step Guide

May 2020
Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

© 2020 Amazon Web Services, Inc. or its affiliates. All rights reserved.
About this Guide

Amazon SageMaker is a fully managed service that provides every developer and data scientist with the ability to build, train, and deploy machine learning (ML) models quickly. Julia is a fast-evolving machine learning and data science programming language, which is increasing in popularity.

This guide outlines the procedure for environment setup and data visualization using Julia within a fully managed Amazon SageMaker Jupyter notebook instance.
Overview

Today, artificial intelligence (AI) technologies are working to be incorporated into products that will shape the future of industries and daily life. Industries, such as manufacturing, finance, and automotive are continually looking for new AI functions to improve customer experiences. The accelerating performance of AI is driven by state-of-the-art machine learning (ML) technologies which require well-managed data modeling, feature engineering, algorithm designing, and ML model training tasks. The efficiency of programming language and toolkit could lead to significant difference of performance in adoption and innovation with AI and ML.

**Jupyter notebook** is the most commonly used working environment for such workloads. Jupyter is a web-based IDE for wide range of workflows in data science, scientific computing, and machine learning.

**Julia**, a programming language, is designed specifically for science computing. It is a high-performance, dynamic, flexible, and easy to use language for scientific and numerical computing. Julia compiles programs to native code via **LLVM**, enabling the performance to be comparable to traditional statically-typed languages.

**Amazon SageMaker** is a fully managed service that is highly flexible and supports both Jupyter notebooks and Julia with full functionality and compatibility. In this guide we will explore how to get started combining these powerful tools and start an exciting journey using Julia as an alternative for data science and machine learning.

This guide will walk you through the following steps:

- Creating an Amazon SageMaker notebook instance
- Creating a Julia environment
- Installing *IJulia* and running a Julia notebook
- Testing the Julia notebook

Before You Begin

This guide assumes that you have the following:

- An AWS account with the capability to create new instances in the Amazon SageMaker console.
- A basic understanding of machine learning or data processing.
Setting Up Your Environment

Creating an Amazon SageMaker Notebook Instance

1. Sign in to Amazon SageMaker console, and open the Amazon SageMaker Dashboard.
2. Select Notebook instances. Alternatively, on the left menu panel, select Notebook instances in the Notebook section.

![Notebook instances button](image)

*Figure 1 – The Notebook instances button on the Amazon SageMaker Dashboard*

3. Select Create notebook instance.

![Create notebook instance button](image)

*Figure 2 – The Create notebook instance button*

4. For Notebook instance name enter the name, and for the Notebook instance type select the instance type for your workload and budget.

Note: The creation of a new notebook instance will take approximately five minutes, based on the instance type selected and the region it is located in. Once provisioned, you can find the provisioned instance ready for use on the Notebook instance screen.
5. Select **Open JupyterLab**, to launch a console of your notebook instance

![Figure 4 – A view of the Notebook instances screen](image)

The launch process may take several seconds, depending on your network and the size of your instance. The JupyterLab console will be open in a pop up window.
Note: The Julia kernel for Jupyter, IJulia, can be used in the Jupyter Notebook and the enhanced alternative, JupyterLab. Please note that the following sections of this guide assume that you are using JupyterLab and not Jupyter.

Creating a Julia Environment

Use the following procedure to create Julia environment in the JupyterLab console.

1. In the JupyterLab console, navigate to the Launcher tab if visible, scroll to the bottom of the tab, and select Terminal.

Figure 5 – A JupyterLab console showing the Launcher tab

Figure 6 – The Terminal icon on the Launcher tab
2. If the Launcher tab is not visible in the JupyterLab console, navigate to the menu bar, and select File, New, Terminal.

3. A new tab, with a Linux shell console attached, will open.

4. To create a new julia conda environment for Julia, and to switch into it, run the following command.

```bash
#!/bin/sh
source activate
conda create --yes -n julia
conda activate julia
```

5. To verify your current environment, run the following command.
#!/bin/sh
conda env list

The expected result is shown below. An environment named julia is marked with an asterisk (*) to indicate that it is currently active.

![Terminal](image.png)

Figure 9 – A Linux shell console showing that the Julia environment is active

6. To install the Julia language package and to verify that installation is complete, run the following command.

    #!/bin/sh
    conda install --yes -c conda-forge Julia=1.0.3
    julia -v

As of the date of publication, the expected output is **julia version 1.0.3**. The Julia package version is 1.0.3 on conda.

7. To verify that Julia is successfully installed, run the following command to launch a Julia REPL console.
You now have successfully set up Julia REPL, and have the full capabilities and features of a Julia runtime environment.

```
#!/bin/sh
julia
```

This guide uses conda to manage the software packages and environments they reside in. This simplifies the installation process and maintenance of the working environment. However, if you wish to build a customized working environment for your business needs, you can do so by installing from pre-built binaries or source code tarballs from the Julia website.

**Installing IJulia and running a Julia Notebook**

Use the following procedure to install the Julia kernel (**IJulia** package) for Jupyter. This will make JupyterLab aware of Julia’s existence, and configure the necessary settings, such as new options for the Julia notebook and kernel.

1. To install **IJulia** and activate the Julia kernel for JupyterLab, run the following command on the Julia REPL console.

```
using Pkg
Pkg.add("IJulia")
using IJulia
jupyterlab(detached=true)
```

You will see the following message, which indicates that **IJulia** is up and running.
2. To launch your first Julia notebook in Amazon SageMaker, on the **Launcher** tab, choose **Julia 1.0.3**.

   ![Screenshot of the Launcher tab showing Julia 1.0.3 icon]

   \[\text{Figure 11 – The Julia icon on the Launcher tab}\]

3. To test native Julia code, enter the following snippets into code cells on the newly created Julia notebook.

   ```julia
   versioninfo()
   \[\sum(x, y) = x + y\]
   \[\sum(2, 3)\]
   ```

   The output shows version information of the current Julia runtime environment, as well as a function definition and invocation with `\[\sum\]` as function name.
Figure 12 – A Julia notebook showing the above expressions entered into the cells and the expected output

Julia Notebook Examples

Use the following procedure to experiment further with how Julia enables you to visualize data.

1. To install the packages Plots and DataFrames, enter the following in to the Julia notebook cells.

```julia
import Pkg
Pkg.add("Plots")
Pkg.add("DataFrames")
```

You will employ Pkg, a built-in package manager, to install Plots and DataFrames, and any required dependent packages will be resolved and installed automatically by Pkg. You should see the following results.
Figure 13 – The Julia notebook output after installation of the Plots and DataFrames packages

2. To generate a sequence of integers and a sequence of random float values, run the following command.

```julia
using Random
A = [1:1000...]
B = [randn(Float64) for i in A]
last(A)
```

The above commands will create two arrays: A containing sequential numbers ranging from 1 to 1000, and B containing 1000 random float values. The expected result is 1000.

3. To create a DataFrame object from the two previously created arrays, run the following command.

```julia
using DataFrames
df = DataFrame(X = A, Y = B)
sort!(df, [:X])
last(df, 5)
```

A DataFrame (df) object is created, where column X contains the values from array A, and column Y contains the values from array B. Then, object df is sorted by the values in column X, and the last five rows are reported as shown below.
5 rows × 2 columns

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>996</td>
<td>1.25086</td>
</tr>
<tr>
<td>2</td>
<td>997</td>
<td>1.11882</td>
</tr>
<tr>
<td>3</td>
<td>998</td>
<td>-0.0898521</td>
</tr>
<tr>
<td>4</td>
<td>999</td>
<td>2.03507</td>
</tr>
<tr>
<td>5</td>
<td>1000</td>
<td>0.0570495</td>
</tr>
</tbody>
</table>

*Figure 14 – The final five rows of df when sorted by the value of column X*

4. To plot the data as a scatterplot, run the following command.

```julia
using Plots
scatter(df.X, df.Y, w=3)
```

*Figure 15 – The data from df plotted on a scatterplot*

5. To plot the data as a histogram, run the following command.

```julia
using Plots
histogram(df.Y, bins=:scott, weights=repeat(1:5, outer=200))
```
Deleting Your Instance

When your notebook instance is no longer in use, we recommend deleting the instance. Use the following procedure to stop and delete your notebook instance.

1. Navigate to your Notebook instance list, and choose the instance that you want to delete.
2. To stop your instance, select **Actions menu, Stop menu**.
   
   The process of stopping an instance may take several minutes. When it is completed, the status column in your Notebook instance list will show as Stopped.
3. Select the instance that was stopped previously, and select **Actions menu**, then **Delete menu**.

Conclusion

Amazon SageMaker is an open and extendible platform which can be integrated with a wide range of tools. This guide walks you through the steps required to set up and run a Julia notebook on Amazon SageMaker. It has also provided some simple examples to test the Julia language and some basic packages, such as Pkg, Plots, and DataFrames. The combination of Amazon SageMaker, Jupyter, and Julia in this guide presented an intuitive feeling of Julia as a programming language, and established a journey for you to explore an interesting and powerful alternative approach in data science and machine learning.
Contributors

Contributors to this document include:

- Laurence MIAO, Solutions Architect, GCR Commercial Sector

Document Revisions

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 2020</td>
<td>First publication</td>
</tr>
</tbody>
</table>