Installation Guide for WCA extension on VSCode

This document provides information on installing software tools with IDE VSCode that are needed to run the **watsonx Code Assistant For Enterprise Java Application** (WCA - EJA) extension. Please note that this extension will interact with a back-end watsonx Code Assistant either installed on the Cloud or within your premises. The Hands-on Labs will be run with the cloud based back-end service.

Last updated: Nov 25th, 2024

1. Java Installation

Install Java 21 using this link:

- Download Java for MacOS - Arm64
- Download Java for MacOS - x86
- Download Java for Windows All the above are compressed files, you can extract them to any folder on your local machine.

Check if Java is installed properly:

```
java --version
```

After installing java, add java to `PATH` variable and set `JAVA_HOME` envitonment variable by follwoing the instructions below:

For Mac:

- Open `.zshrc` or `.bash_profile`

  ```
  nano ~/.zshrc
  ```

- Add the following lines:

  ```
  export JAVA_HOME=/Library/Java/JavaVirtualMachines/<java
  version>/Contents/Home
  ```

  ```
  export PATH=$JAVA_HOME/bin:$PATH
  ```

- Save the file and exit (press CTRL + X, then Y, and hit Enter)
- Reload the shell configuration so the changes take effect.

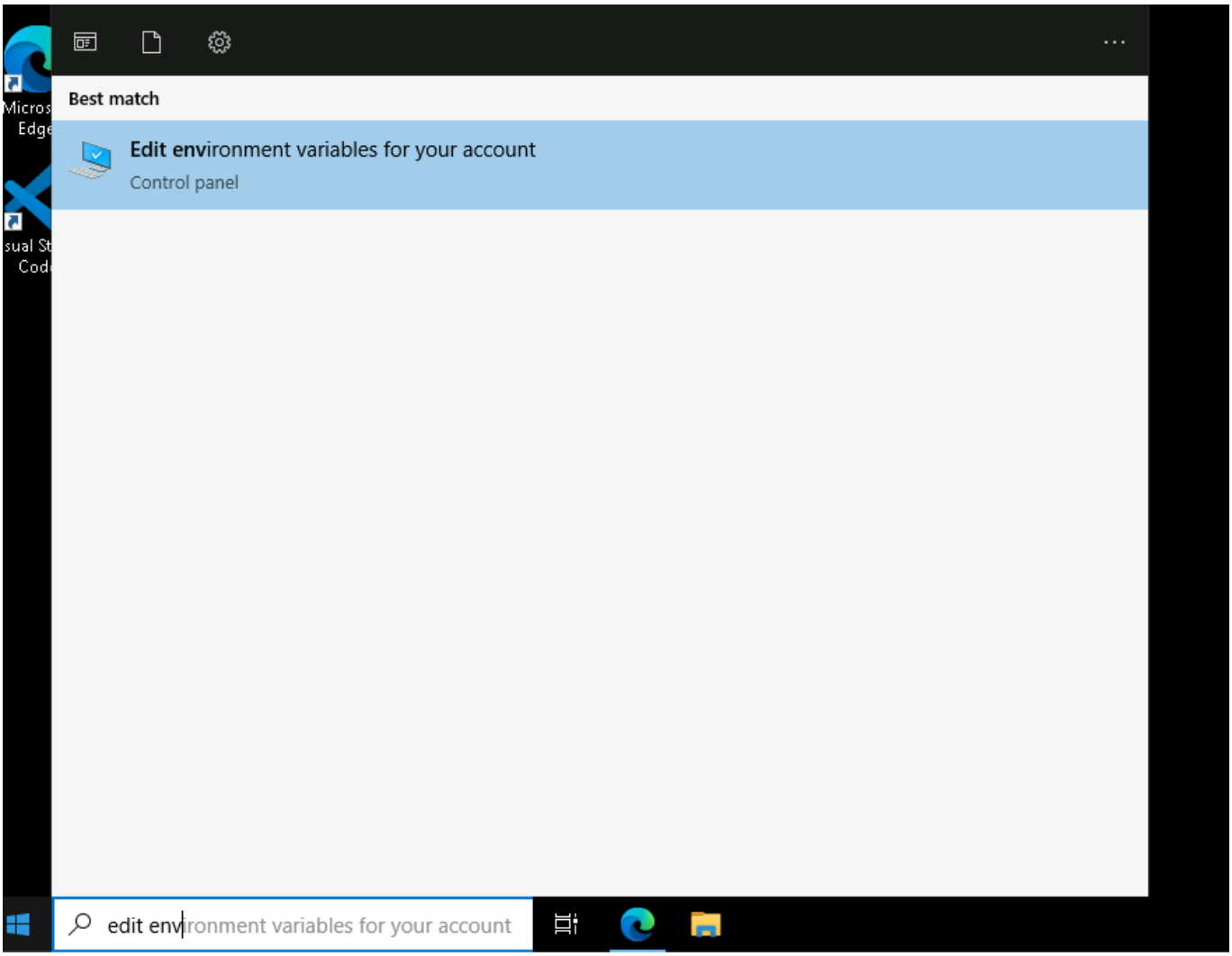
```
source ~/.zshrc
```

- Check your `JAVA_HOME` environment variable with the following command:

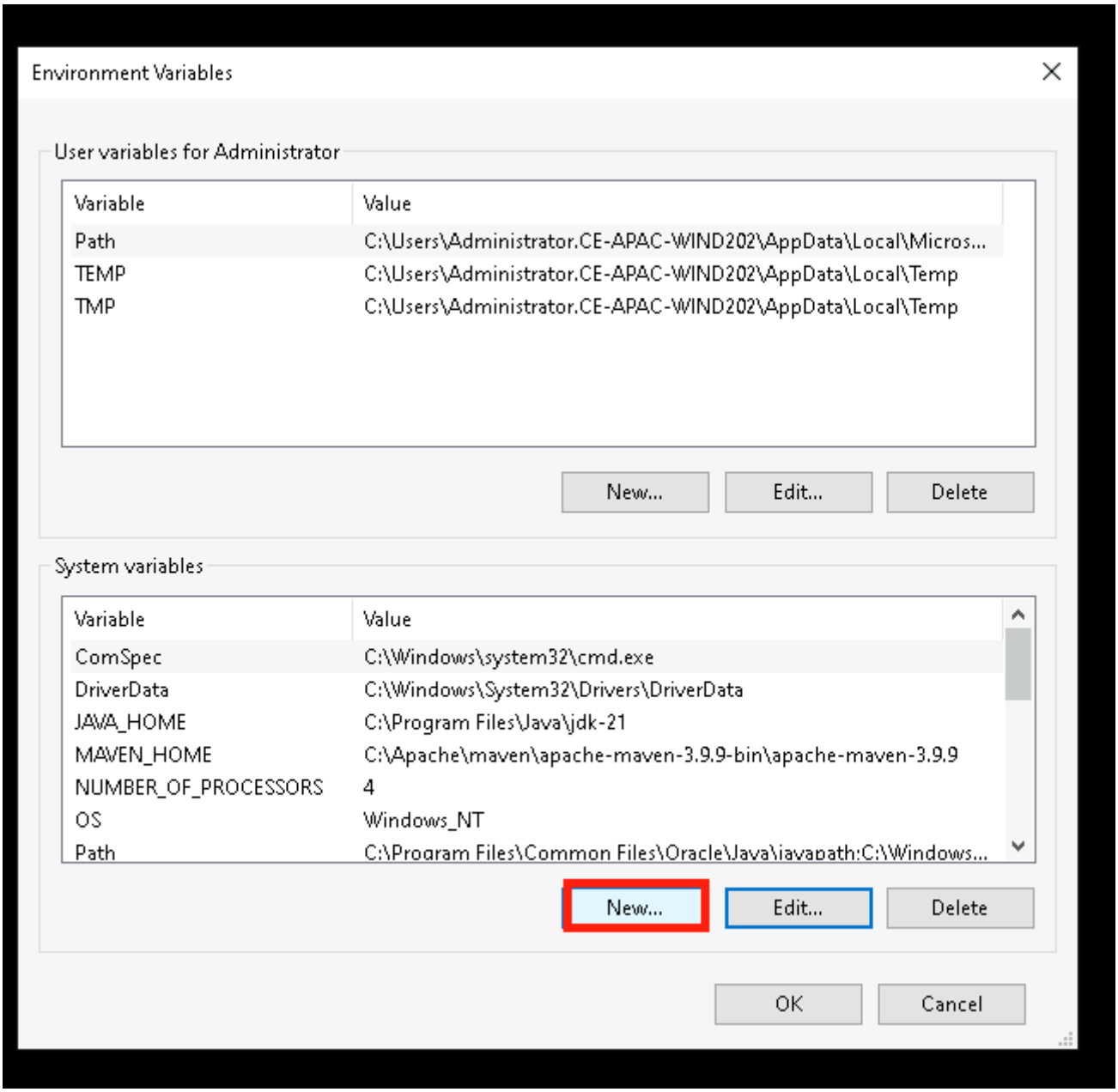
```
echo $JAVA_HOME
```

For Windows:

- Open Environment variables using the Windows search bar (search for "edit environment variables" in the search bar).

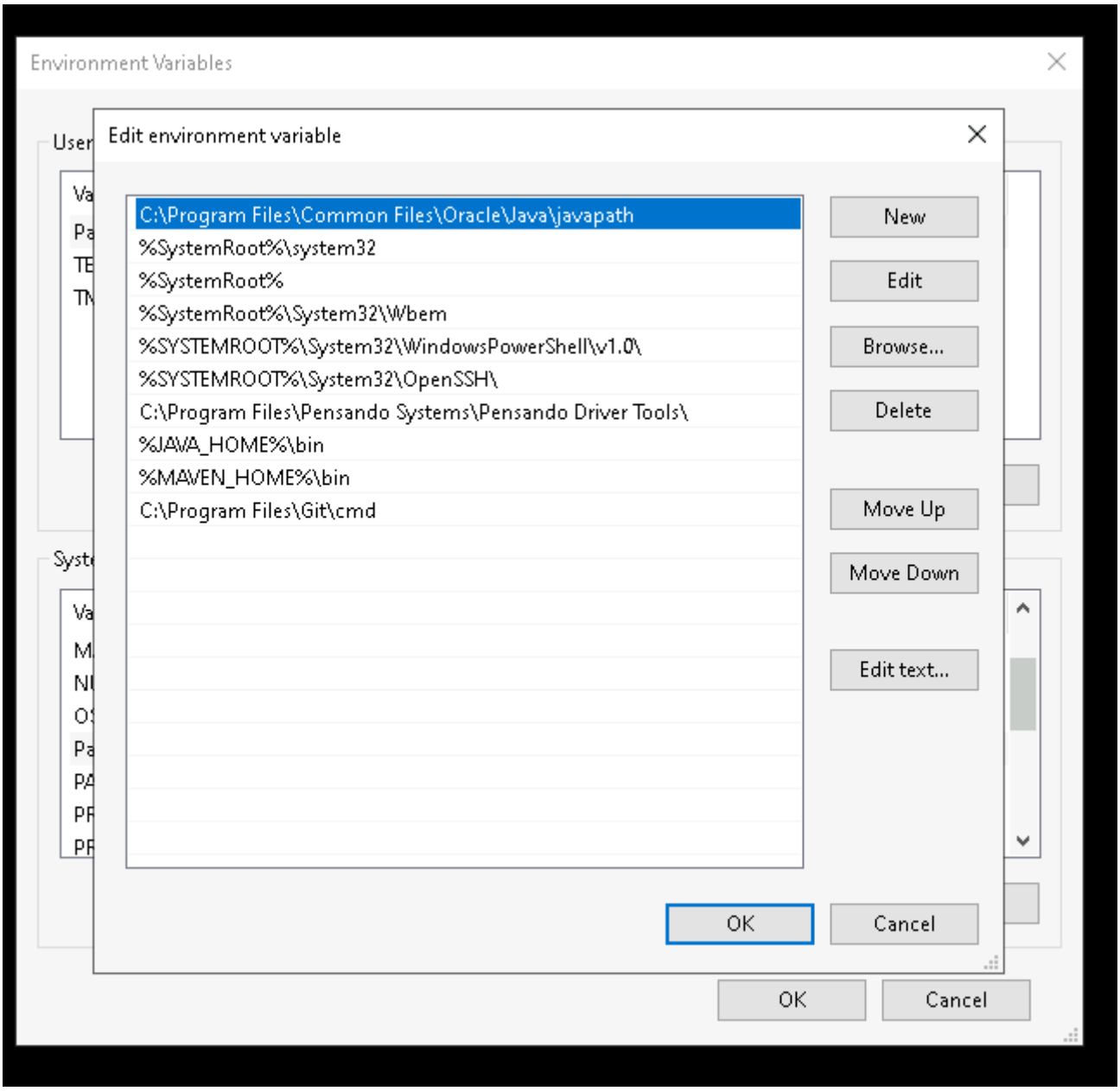


- Set the `JAVA_HOME` variable using Environment variables (click on "New" if you do not have a `JAVA_HOME` set, or click on "Edit" to change the existing `JAVA_HOME`, and point it to the Java you installed in the earlier steps):




```
JAVA_HOME=C:\Program Files\Java\jdk-21
```

- Add Java to your PATH using environment variables:




```
%JAVA_HOME%\bin
```

2. Install Maven

For Mac:

- Install Maven using Homebrew:

```
brew install maven
```

- Check that maven is installed properly:

```
mvn --version
```

For Windows:

- Visit the official Maven website: Maven Download Page
- Under "Files", click on the binary zip archive link (e.g., apache-maven-x.x.x-bin.zip).
- Extract the zip file to a location of your choice, e.g., C:\Apache\maven.
- Set the `MAVEN_HOME` variable using Environment variables:

```
MAVEN_HOME=C:\Apache\maven\apache-maven-3.9.9
```

- Add Maven to your `PATH` using Environment variables:

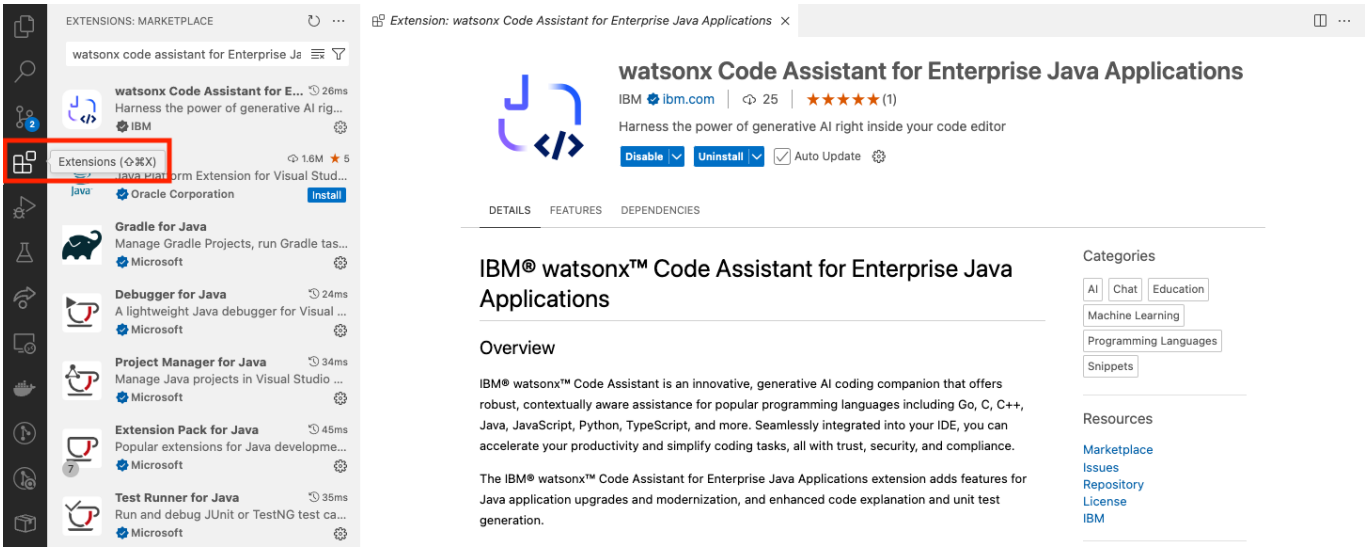
```
<path-to-folder>\maven\apache-maven-3.9.9-bin\apache-maven-3.9.9\bin
```

3. Install VSCode

Visit the VSCode Official Website for installation.

4. Download the WCA - EJA Extension

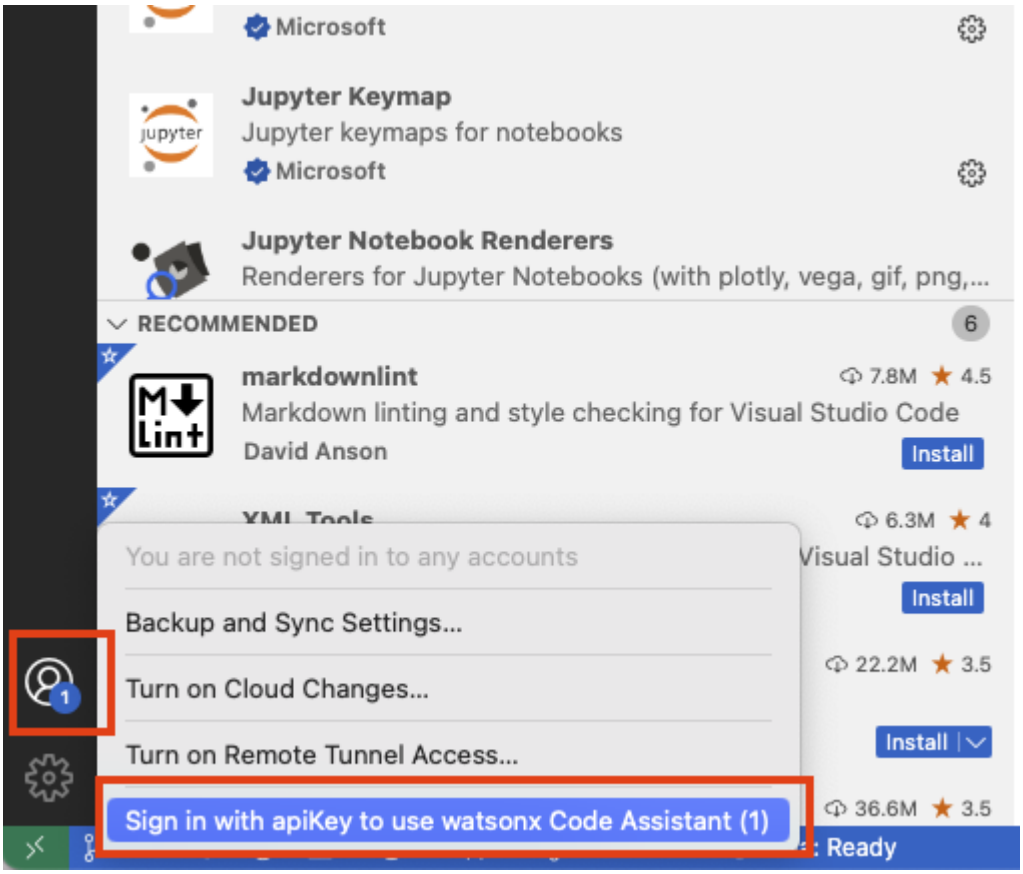
Download the latest '**watsonx Code Assistant for Enterprise Java Applications**' VSCode Extension from the marketplace.

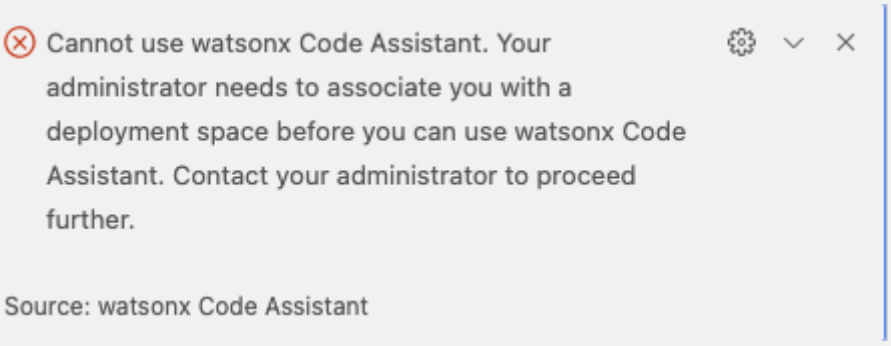


> **Note:** You will notice that 2 VSCode extensions are automatically downloaded. This is because the '**watsonx Code Assistant for Enterprise Java Applications**' extension depends on some services provided by the core '**watsonx Code Assistant**' extension.

5. Setting up WCA - EJA Extension

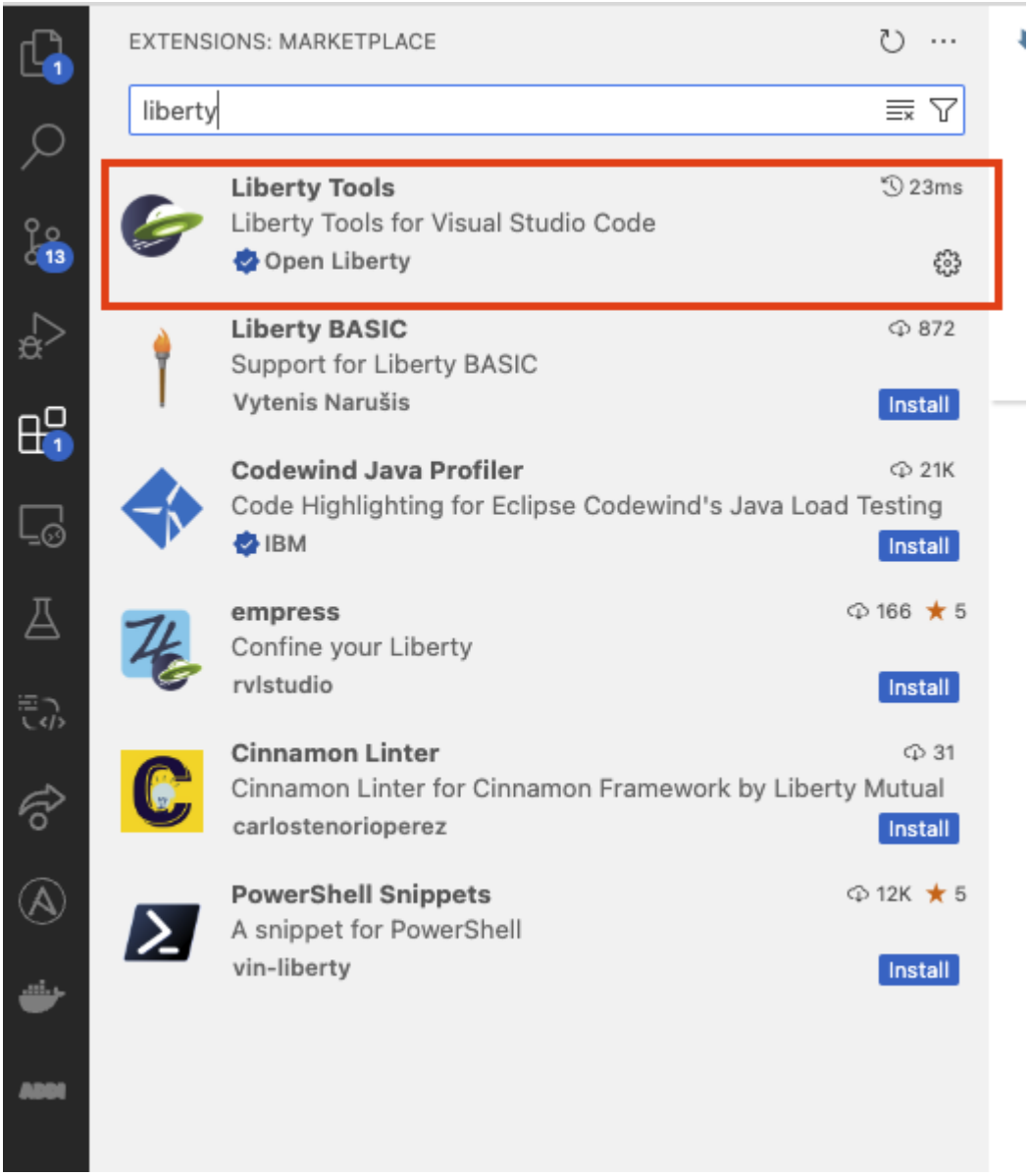
Login with the WCA - EJA API Key at the bottom left corner of VSCode. After successfully signing in, the number indicator should be gone.

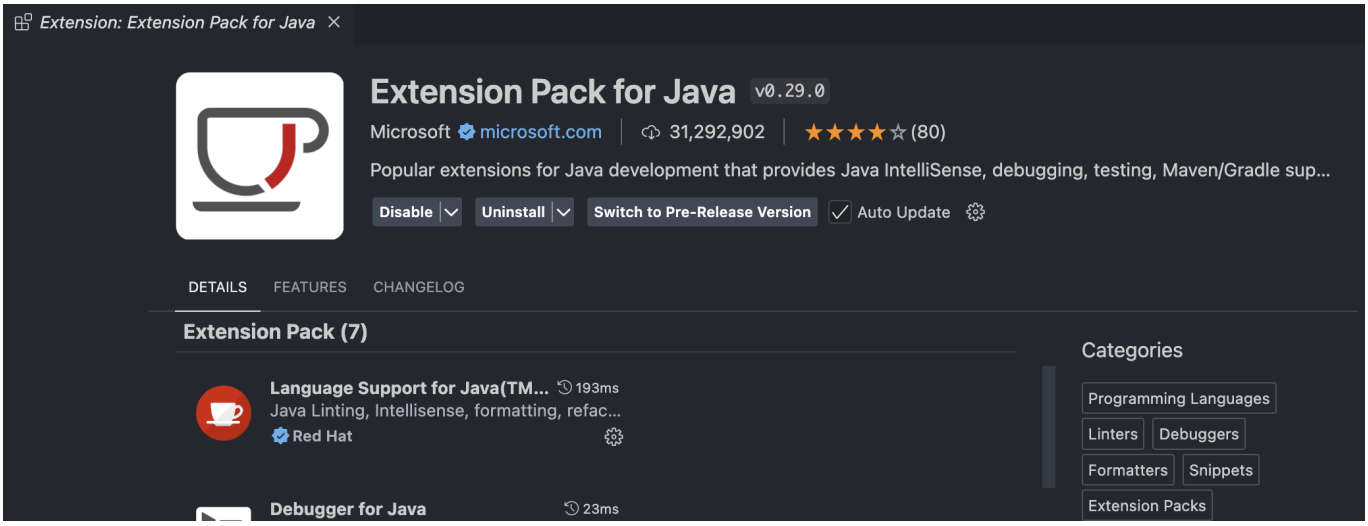


> **Note:** If you encounter an issue during authorization that says **"administrator needs to associate you with a deployment space"**, please reach out to IBMers to set up the deployment space again
>
> 
>
>
> for your API Key.

6. Installing Liberty Tools and Java Extension

Install the Liberty Tools and Extension Pack for Java extensions from the VSCode marketplace as shown below.





7. Connectivity test to IBM Cloud

To be able to run the hands-on labs from your workstation, you will need to be able to reach the **watsonx Code Assistant** SaaS instance which is hosted in our Dallas datacenter in the US South region. To validate that you are able to reach it by running the following commands in a terminal (output should be similar to the following):

```
$ ping us-south.ml.cloud.ibm.com
```

The ouptut should look like the following (you can stop it after a few packets by pressing `CTRL+C`):

```
$ ping us-south.ml.cloud.ibm.com
PING us-south.ml.cloud.ibm.com (104.17.182.188): 56 data bytes
64 bytes from 104.17.182.188: icmp_seq=0 ttl=52 time=29.392 ms
64 bytes from 104.17.182.188: icmp_seq=1 ttl=52 time=25.007 ms
^C
--- us-south.ml.cloud.ibm.com ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 25.007/27.200/29.392/2.192 ms
```

Then test the second endpoint of IBM Cloud Identity and Access Management:

```
$ ping iam.cloud.ibm.com
```

The ouptut should look like the following (you can stop it after a few packets by pressing `CTRL+C`):

```
(base)
❯ ping
PING iam.cloud.ibm.com (23.192.118.8): 56 data bytes
64 bytes from 23.192.118.8: icmp_seq=0 ttl=58 time=21.100 ms
64 bytes from 23.192.118.8: icmp_seq=1 ttl=58 time=18.565 ms
^C
--- iam.cloud.ibm.com ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
```

Note: if you see that packets are not being transmitted, reach out to an IBMer (preferrably before the workshop) to find a backup option for you to run the labs.

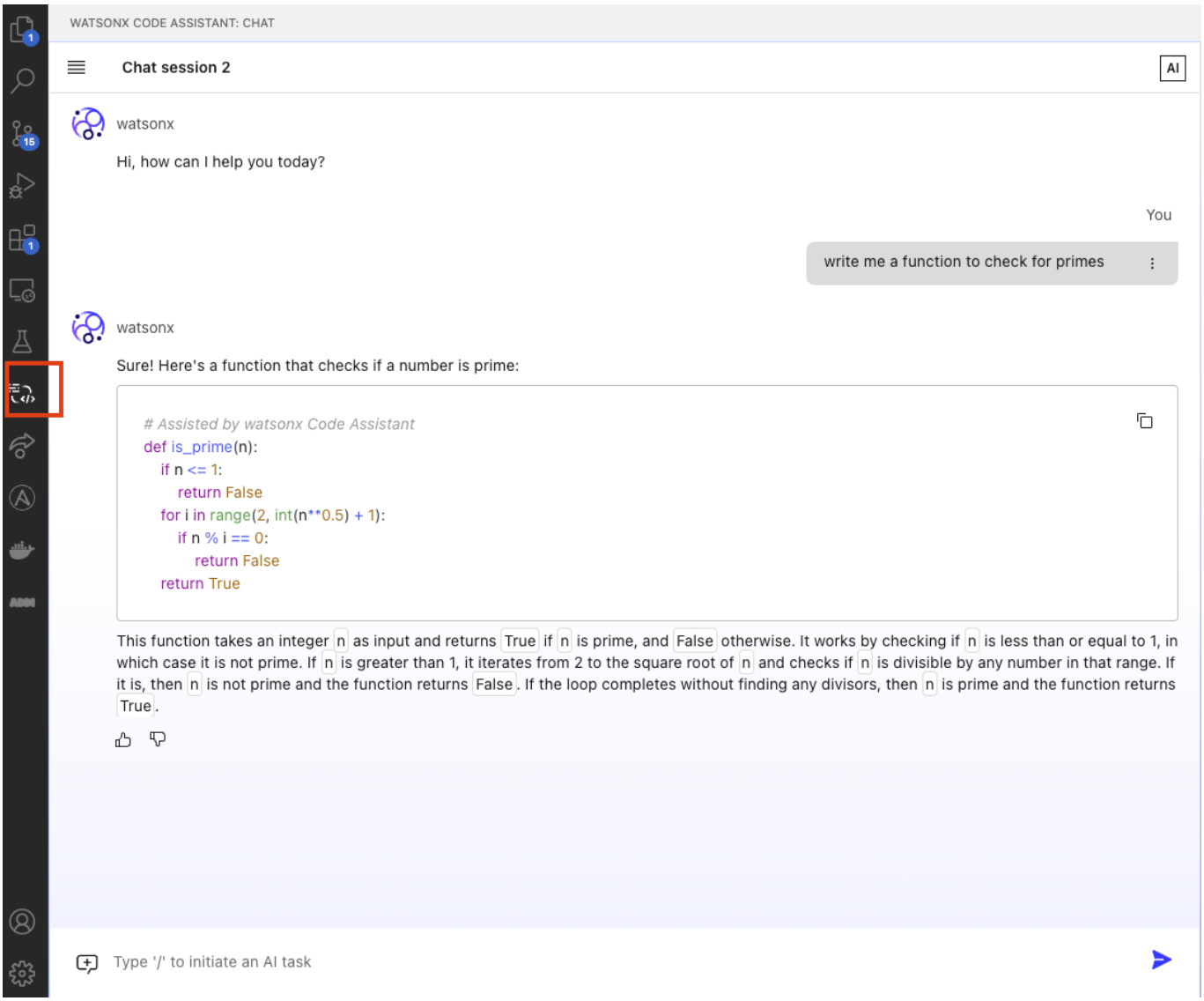
8. WCA - EJA API Key

The VSCode extension requires a connection to a **watsonx Code Assistant** SaaS service. The authentication to the backend service can be completed using an API key.

As of now, the API Key will be provided by IBMers. Please reach out to IBMers for help on this.

9. Start Using watsonx Code Assistant

You can check if your API Key is set up correctly by navigating to the **watsonx Code Assistant** tab and opening the chat window to chat with the model.


WATSONX CODE ASSISTANT: CHAT
Chat session 2
AI
watsonx
Hi, how can I help you today?
You
write me a function to check for primes
watsonx
Sure! Here's a function that checks if a number is prime:
# Assisted by watsonx Code Assistant
def is_prime(n):
if n <= 1:
return False
for i in range(2, int(n**0.5) + 1):
if n % i == 0:
return False
return True
This function takes an integer n as input and returns True if n is prime, and False otherwise. It works by checking if n is less than or equal to 1, in which case it is not prime. If n is greater than 1, it iterates from 2 to the square root of n and checks if n is divisible by any number in that range. If it is, then n is not prime and the function returns False. If the loop completes without finding any divisors, then n is prime and the function returns True.
Type '/' to initiate an AI task