

# Modern, Buildable Projects

## with IBM i Project Explorer and Bob

Edmund Reinhardt

Product Architect - IBM i Application Development

[edmund.reinhardt@ca.ibm.com](mailto:edmund.reinhardt@ca.ibm.com)

Sanjula Ganepola

Software Developer

[sanjula.ganepola@ibm.com](mailto:sanjula.ganepola@ibm.com)



# Agenda

- Challenges with Building on IBM I
- How do IBM i Projects and Bob overcome this?
- Ins and Outs of IBM i Project Explorer
- Demo

# Challenges with Building on IBM i

# Building on IBM i is hard...

- 1 SRC-PF
  - 10 char names
  - Fixed record length
  - Not accessible to open ecosystem, including Git and Make
  - Source of the same type stored in QxxxSRC to avoid name conflicts (member type does not disambiguate)
- 2 Libraries
  - Only 2 level hierarchy to organize, with only short 10 char names
- 3 Source control
  - None (sequence number dates)
  - Home grown
  - Proprietary IBM i systems
    - Cost
    - Smaller market = less investment
- 4 Build system
  - Individual CRTXXXMOD + CRTPGM
  - CL Scripts
  - A couple of vendors have dependency-based build

# **How do IBM i Projects and Bob overcome this?**

# Let's use a different (but similar) file system

## MYPROJECT

- QRPGLSRC
  - PROGRAMA.RPGLE
  - PROGRAMB.RPGLE
  - PROGRAMC.RPGLE
- QSQLSRC
  - CUSTOMERS.SQL
  - INVENTORY.SQL
- QCLLESRC
  - START.CLLE
- QCMDSRC
  - STARTJOB.CMD

***QSYS.LIB Library***

No more character  
name restrictions

Now usable with  
Git and Make

Flexible directory  
structure

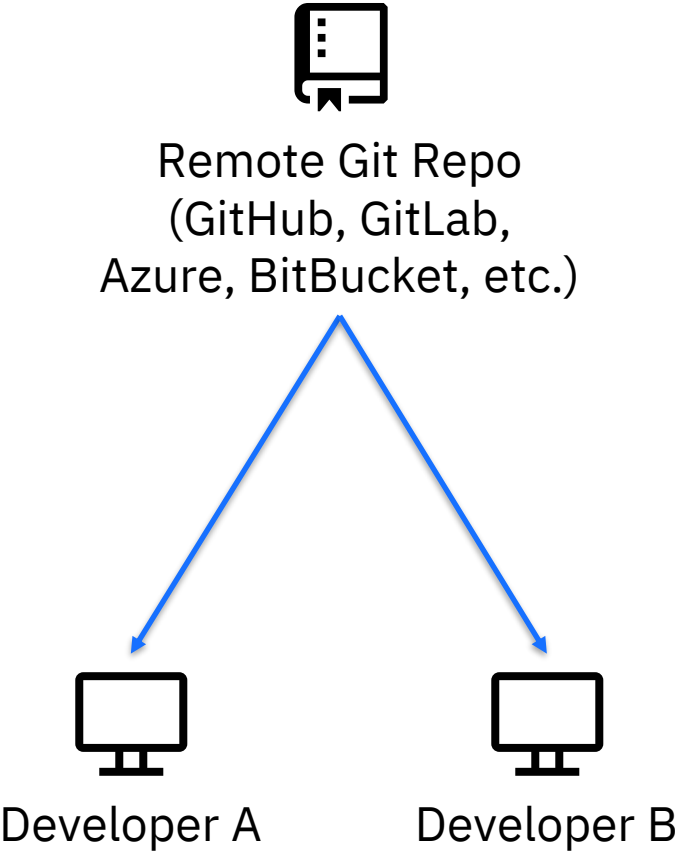
## /my-project

- /.git
- qrpglesrc
  - programa.rpgle
  - programb.rpgle
  - programc.rpgle
- qsqlsrc
  - customers.sql
  - inventory.sql
- qcllesrc
  - start.clle
- qcmdsrc
  - Startjob.cmd

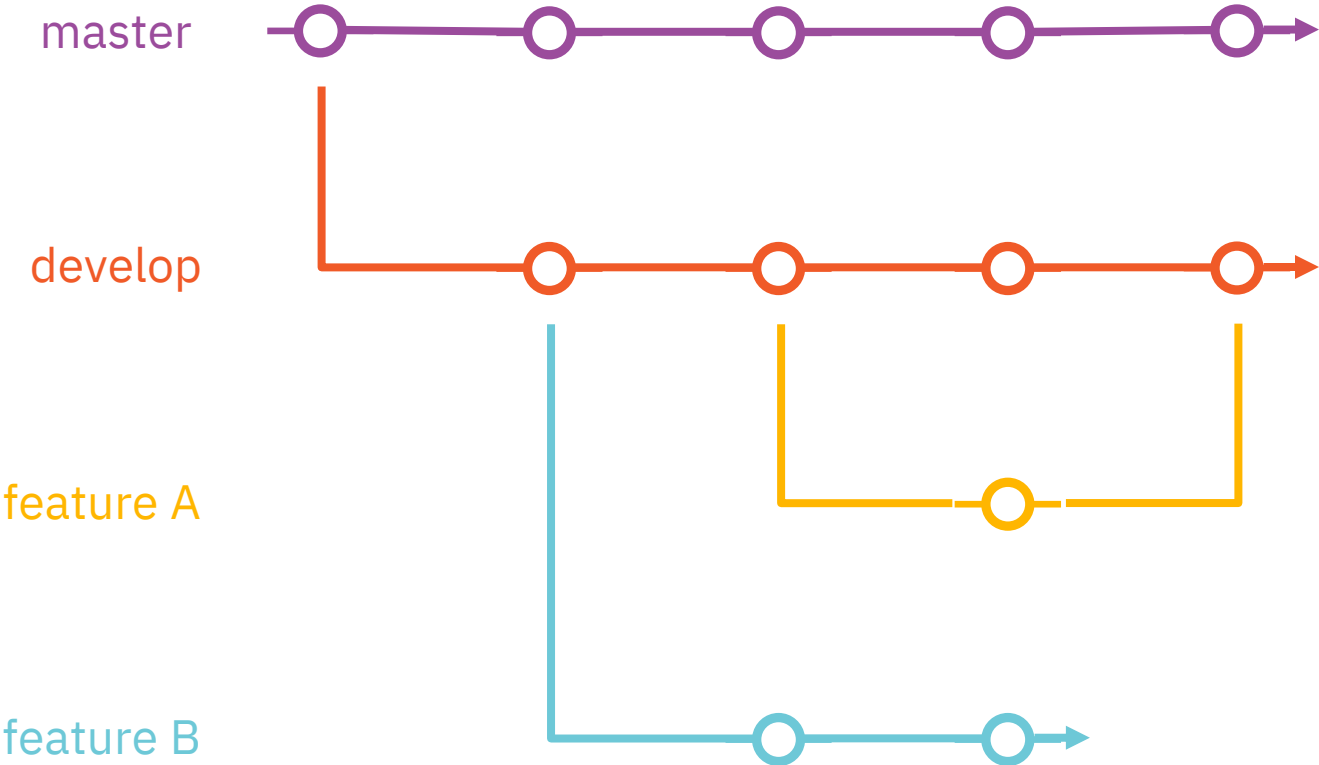
***IFS/Local File System***

# Unlocking source control with Git

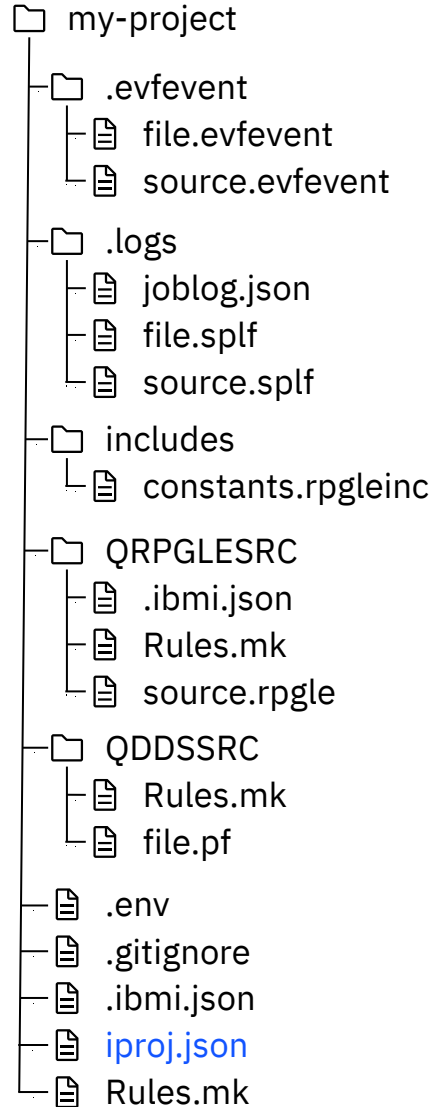
## Distributed Development



## Version Control and Git Workflow



# Projects that self-describe how to build themselves!?



Project  
Information

Configure  
library list

Configure  
build/compile  
environment

```
{ } iproj.json x
{ } iproj.json > ...
1 {
2   "version": "0.0.1",
3   "description": "SAMPLE PROJECT",
4   "repository": "https://github.com/edmundreinhardt/bob-recursive-example.git",
5   "license": "Apache 2.0",
6   "objlib": "&CURLIB",
7   "curlib": "&CURLIB",
8   "includePath": [
9     "includes",
10    "QPROTOSRC"
11  ],
12  "preUsrlib1": [
13    "&lib1"
14  ],
15  "postUsrlib1": [
16    "&lib2"
17  ],
18  "setIBMiEnvCmd": [],
19  "compileCommand": "makei c -f {filename}",
20  "buildCommand": "makei build"
21 }
```

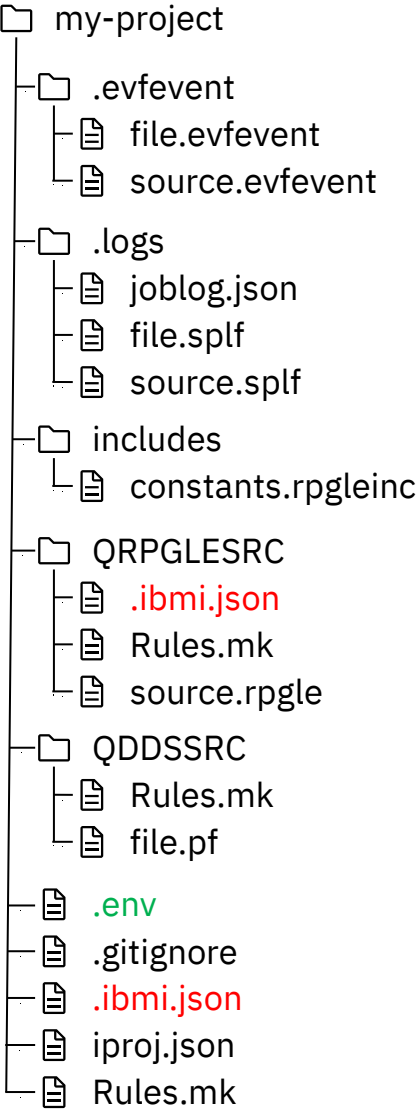
Standardized  
metadata format  
with variables (&...)

Set  
build/compile  
command

*iproj.json in project root*



# Flexible subdirectories and build customization



```
{ } .ibmi.json X
{ } .ibmi.json > ...
1 {
2   "version": "0.0.1",
3   "build": {
4     "tgtCcsid": "273",
5     "objlib": "&lib3"
6   }
7 }
```

*.ibmi.json in subdirectories*

Target object library for directory

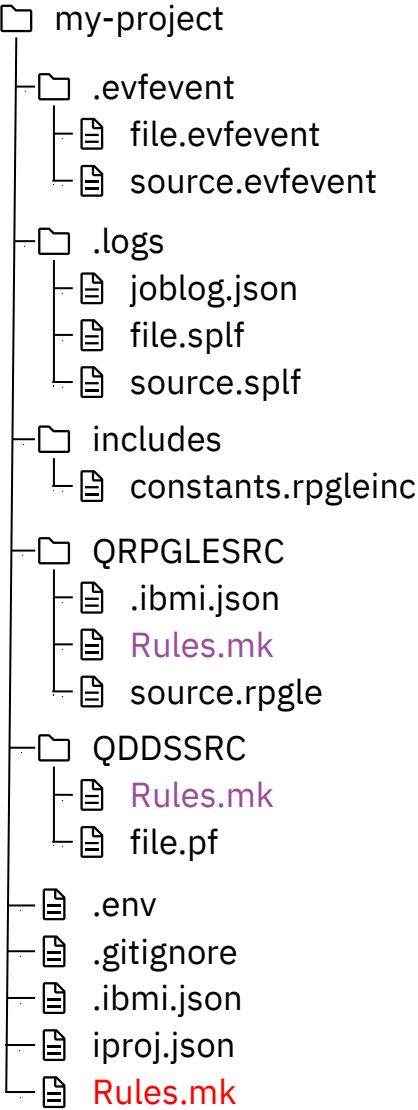
EBCDIC encoding for compiler

```
gear .env X
gear .env
1 LIBL=QGPL QTEMP QDEVELOP QBLDSYS QBLDSYSR
2 CURLIB=SANJULA
3 lib1=MYLIB
4 lib2=ABCLIB
5 lib3=APILIB
```

*.env in project root*

Custom variable values so that each developer can customize build

# Control what objects to build and how to build them



```
M Rules.mk x
M Rules.mk
1 SUBDIRS = qrpglesrc qddssrc
```

*Rules.mk in project root*

Declare subdirectories  
to be built

```
M Rules.mk x
M Rules.mk
1 FVAT.SRVPGM: fvat.bnd VAT300.MODULE
2 FVAT.SRVPGM: TEXT = Functions VAT
3 FVAT.SRVPGM: private TEXT = Functions VAT
4
5 VAT300.MODULE: vat300.rpgle QPROTOSRC/vat.rpgleinc VATDEF.FILE
6 VAT300.MODULE: private TEXT := bound into FVAT.SRVPGM
7 VAT300.MODULE: private DBGVIEW := *SOURCE
8
9 VATDEF.FILE: vatdef.pf SAMREF.FILE
```

*Rules.mk in subdirectories*

Make style list of  
objects to be built and  
from what source files

# Build and Compile Process



## Initialization and Migration

Command	Description
makei init	Create iproj.json
makei cvtsrcpf	Convert QSYS members to Unicode IFS stream files

## Building

Command	Description
makei build	Build the entire project
makei b -t <object>	Build target object
makei b -d <directory>	Build all objects in the specified directory (based on Rules.mk)

## Compiling

Command	Description
makei compile -f <stream file>	Compile target object of specified stream file
makei compile -files file1: file2: ...	Compile target objects of all specified stream files

# **Ins and Outs of IBM i Project Explorer**

# Overview

*The ultimate tool for local development on IBM i!*

Set variables

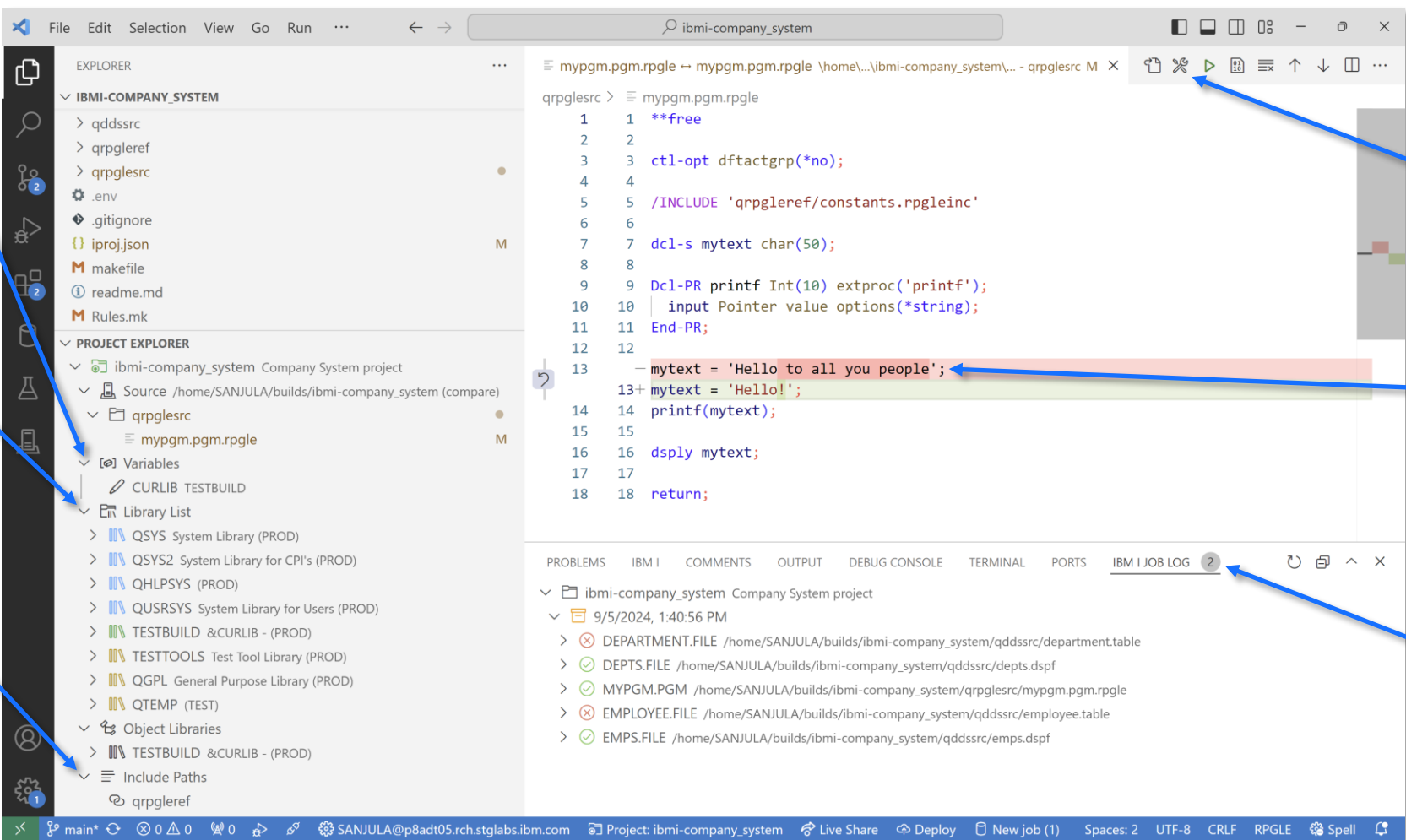
Manage library list

Modify include paths

Build and Compile

Local source vs. IFS source

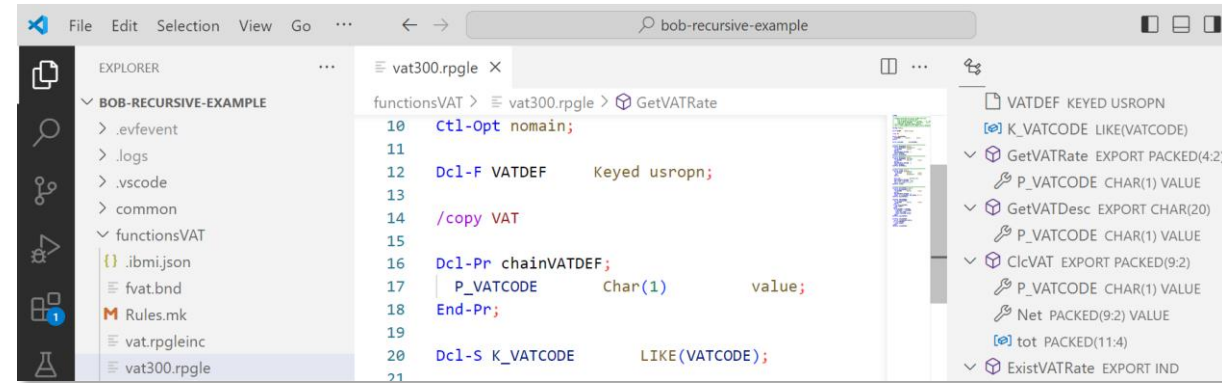
View job logs



# Installation

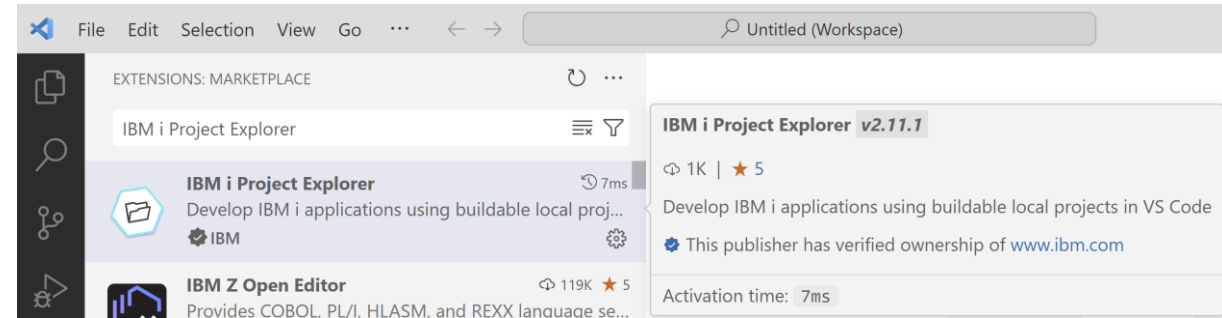
1

*Download  
Visual Studio Code*



2

*Download  
IBM i Project Explorer  
and  
Code for IBM i*



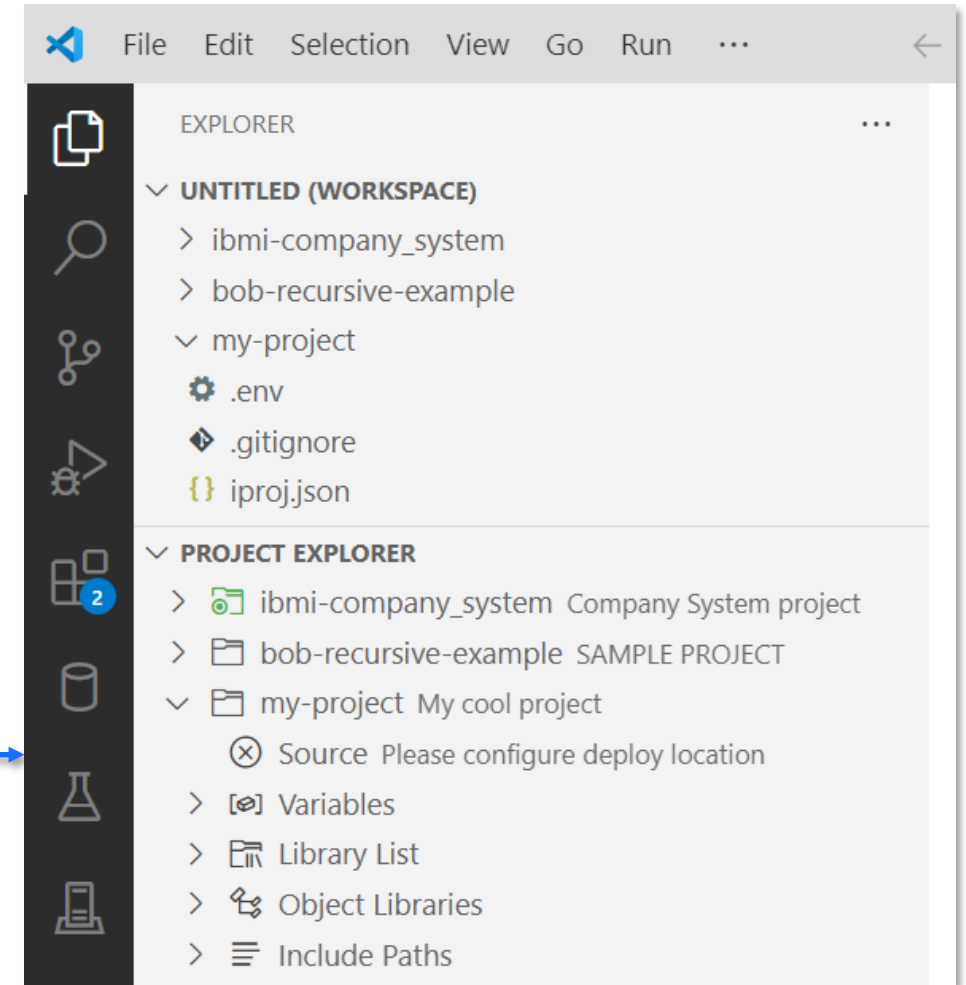
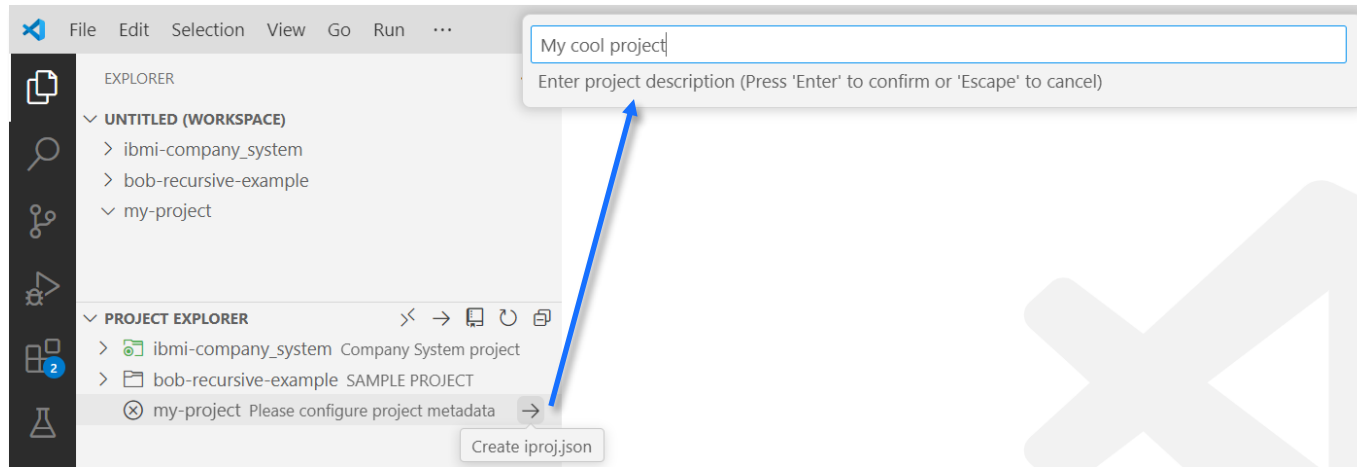
3

*Run  
yum install bob  
on IBM i*



# Create a New Project

- Create and open a folder for your project
- Create an `iproj.json`
- Set the project description
- Connect to an IBM i (using Code for IBM i)



# Migrate Source from QSYS

CVTSRCPF  
from BOB

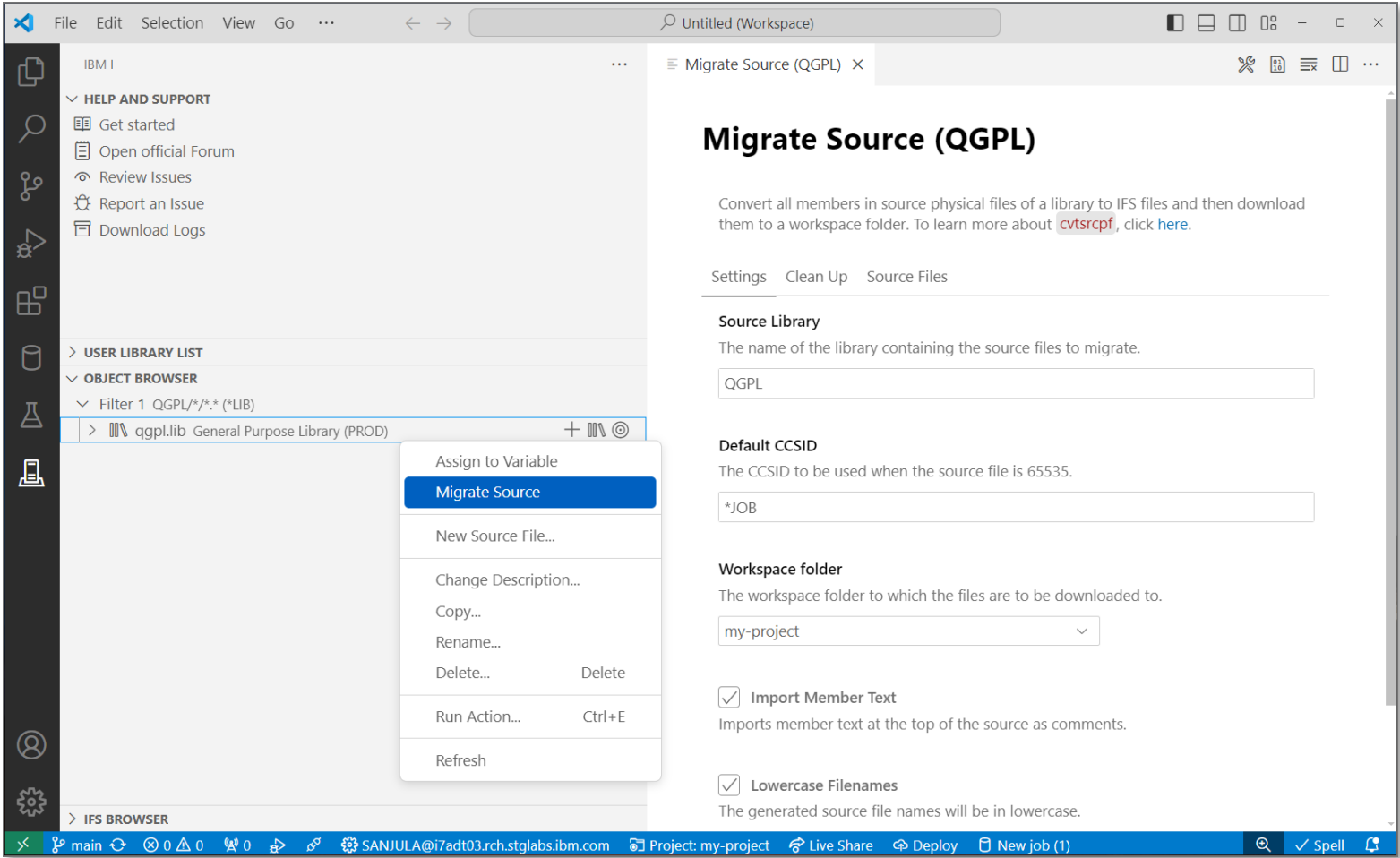


QSYS members in  
source physical files  
↓  
Properly encoded,  
terminated, and named  
source files in an IFS  
directory

↓  
Download to local  
project

↓  
Rename extensions  
↓  
Convert includes/copy  
directives to Unix style  
paths

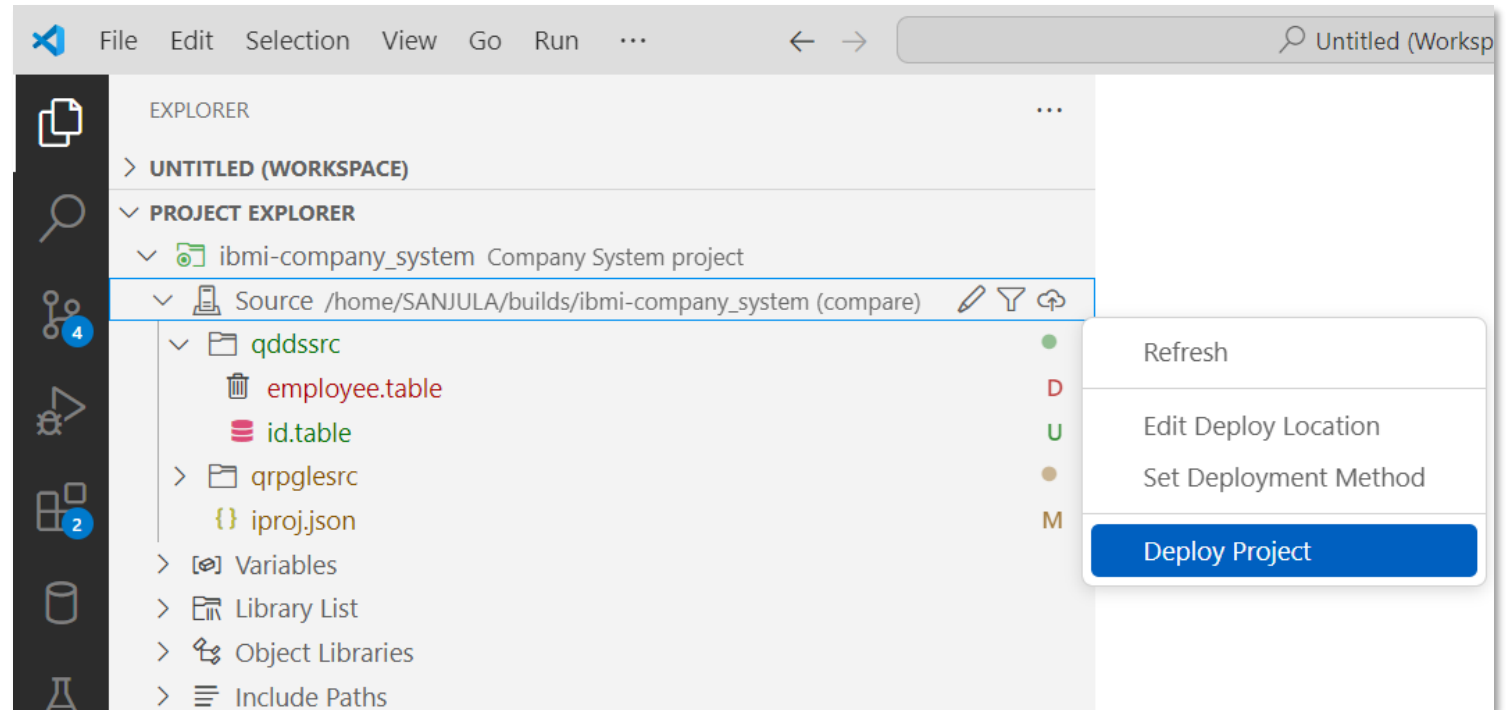
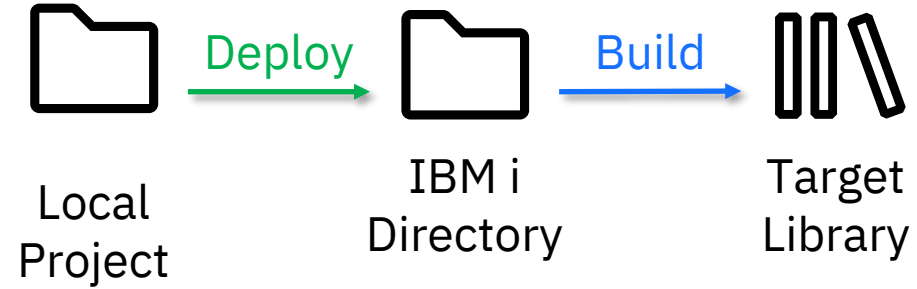
Source Orbit





# Source and Deployment

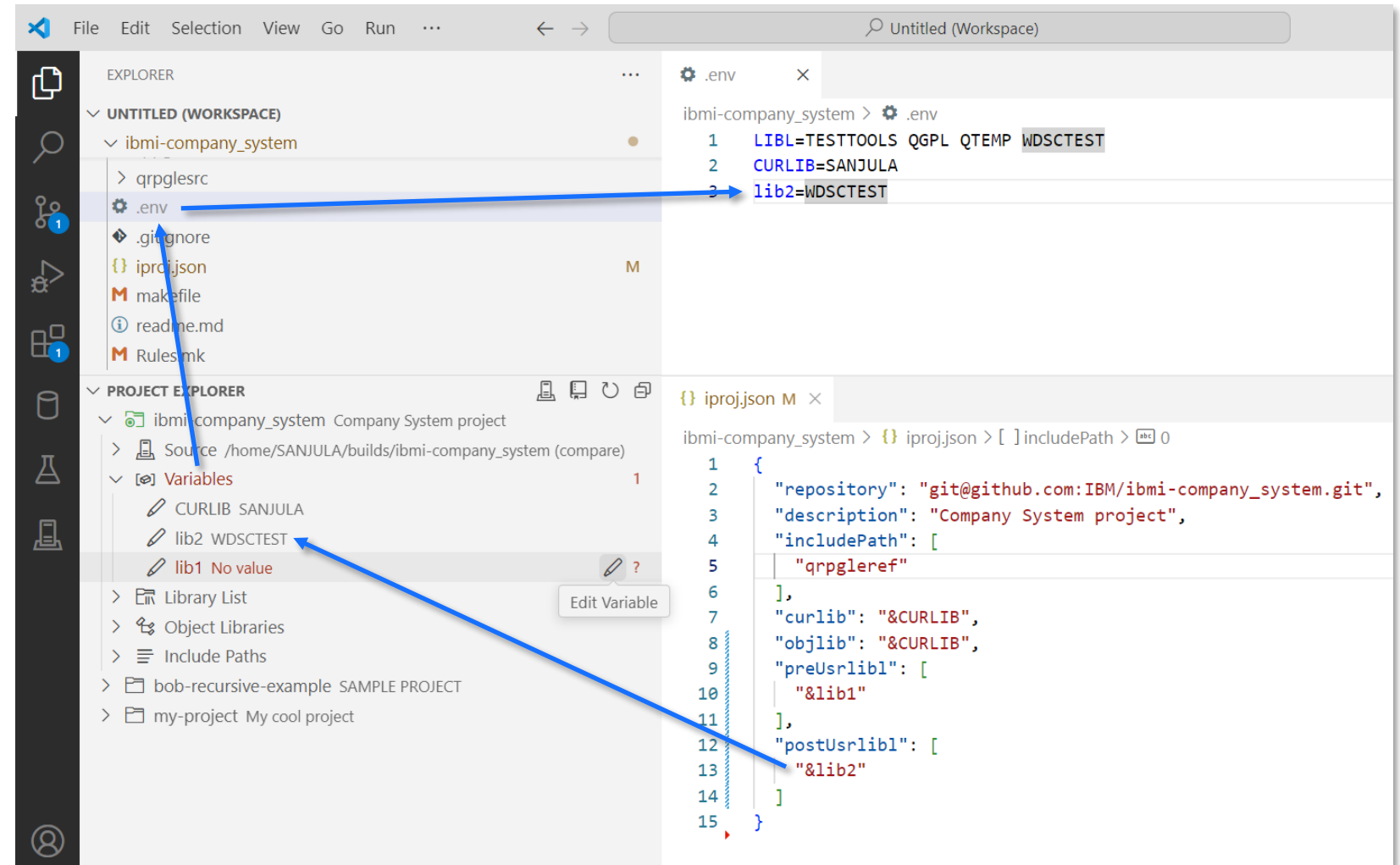
- Set deploy location
  - Where source gets uploaded to
  - Typically set one
  - Each developer gets a unique location
  - Each repository gets a unique location
- Set deployment method
  - Compare (typically the safest)
  - Changes
  - Working Changes
  - Staged Changes
  - All
- Deploy project
  - Moves files to deploy location based on deployment method



# Work with Variables

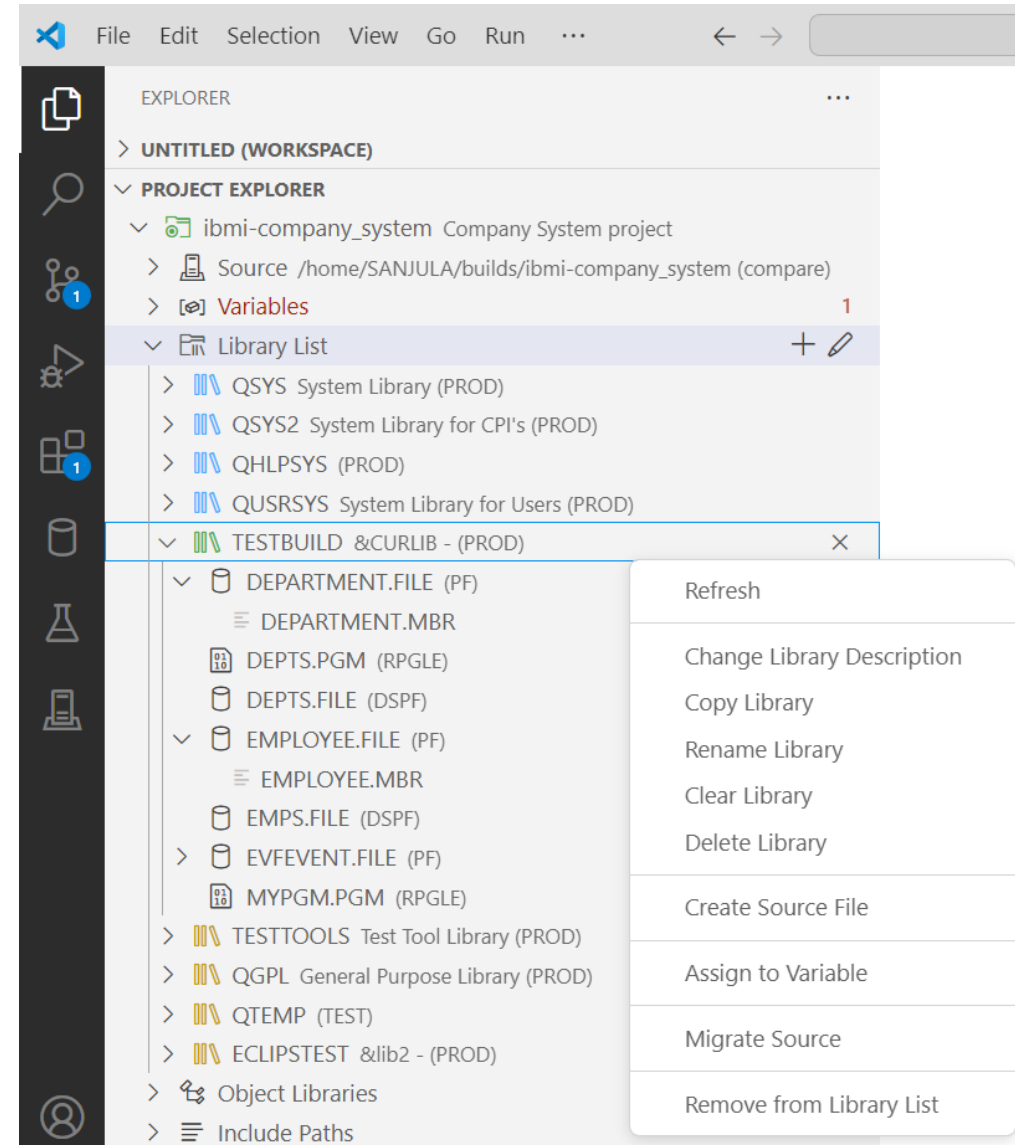
- View and set variables (for libraries, include paths, or build/compile commands)
- Browse for libraries and assign values to variables
- Configure hardcoded values as variables

**Do not push .env file to Git!**



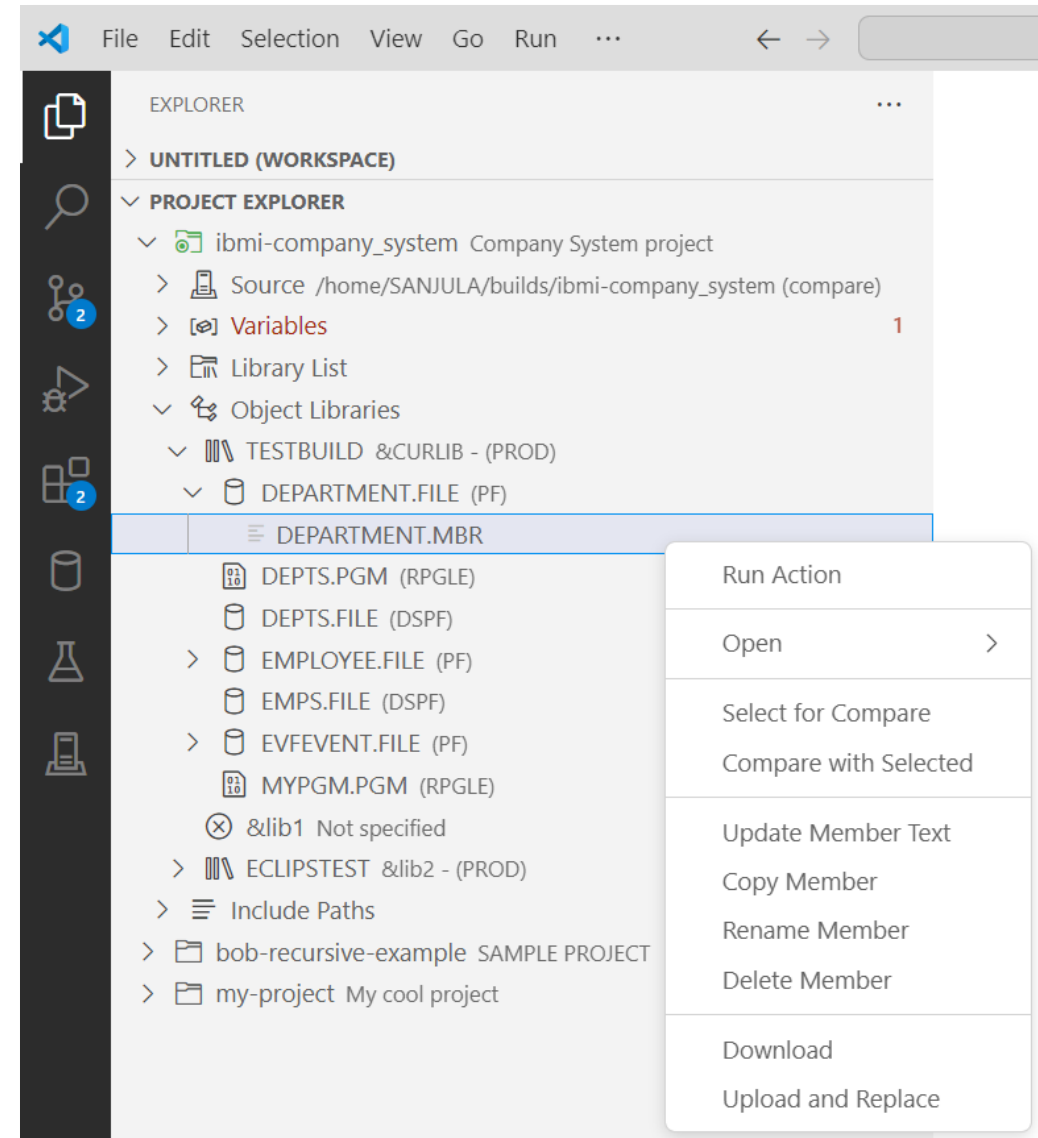
# Manage the Library List

- Add to beginning/end of library list (preUsrlibl and postUsrlibl) and set current library (curlib in iproj.json)
- Reorder library list
- Browse objects and members
- Manage libraries, objects, and members



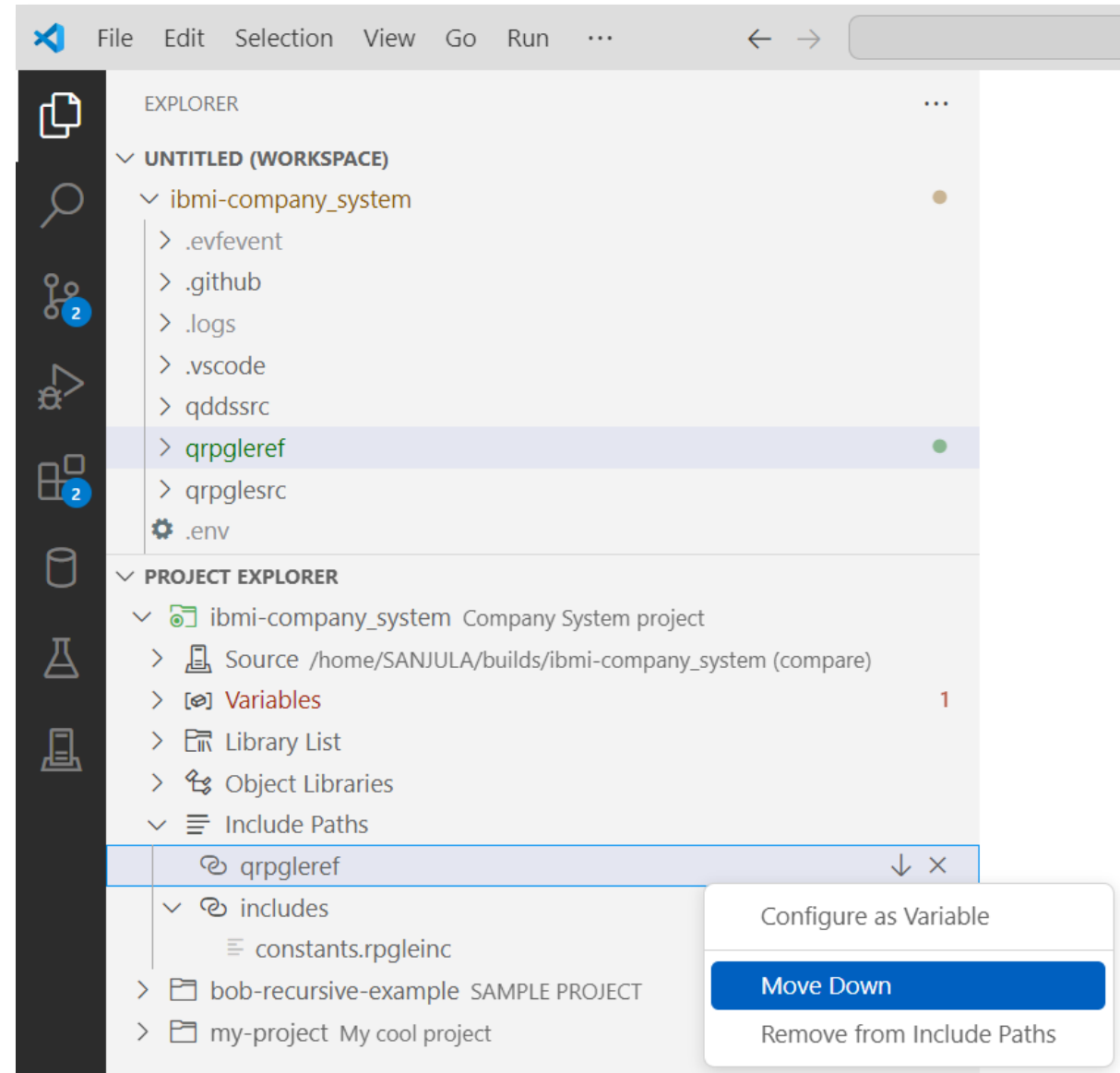
# Browse Object Libraries

- Another place to manage libraries in iproj.json (curlib, objlib, preUsrLibl, postUsrLibl)
- Manage libraries, objects, and members



# Manage Include Paths

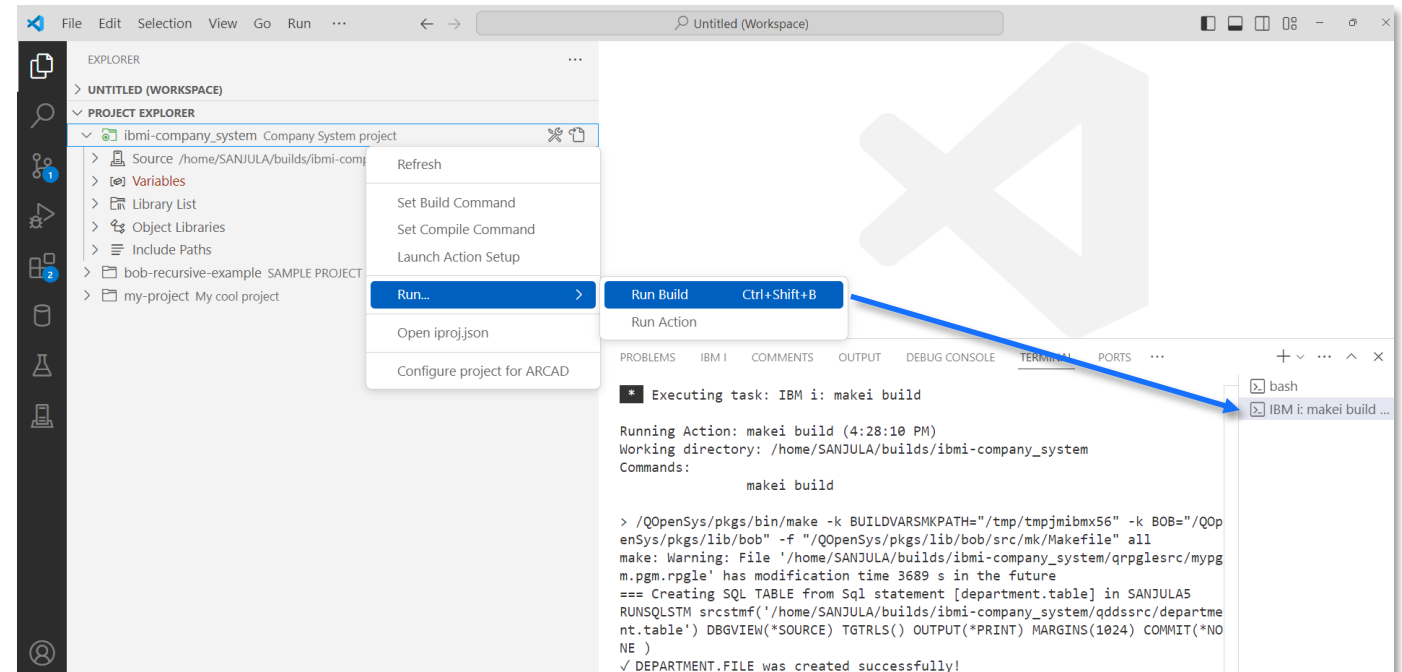
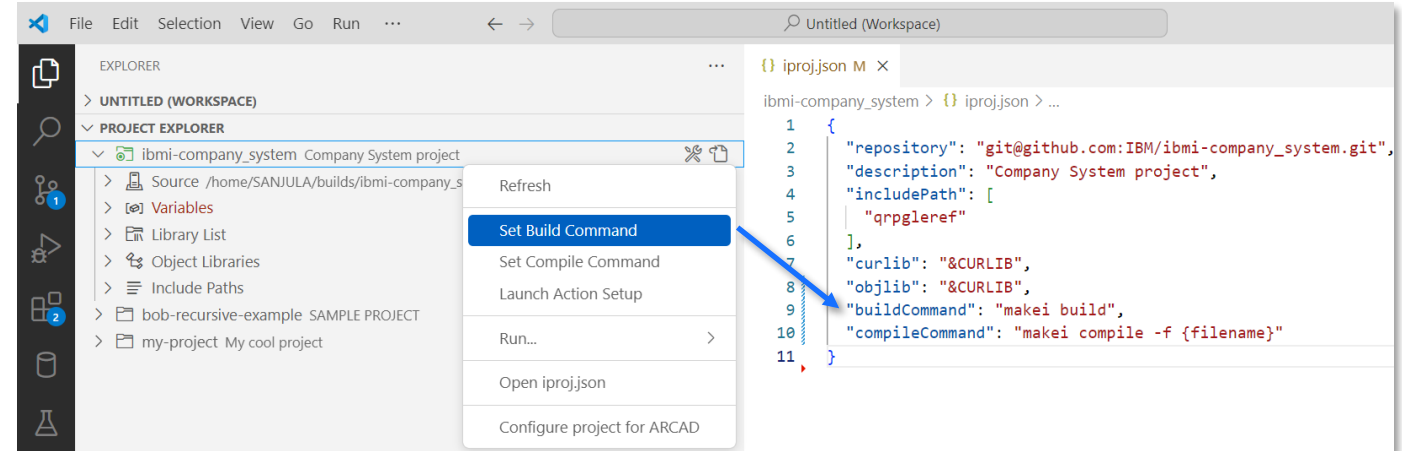
- Add, remove, and reorder include paths
- Visualize if includes resolve locally or to remote IFS



# Build and Compile

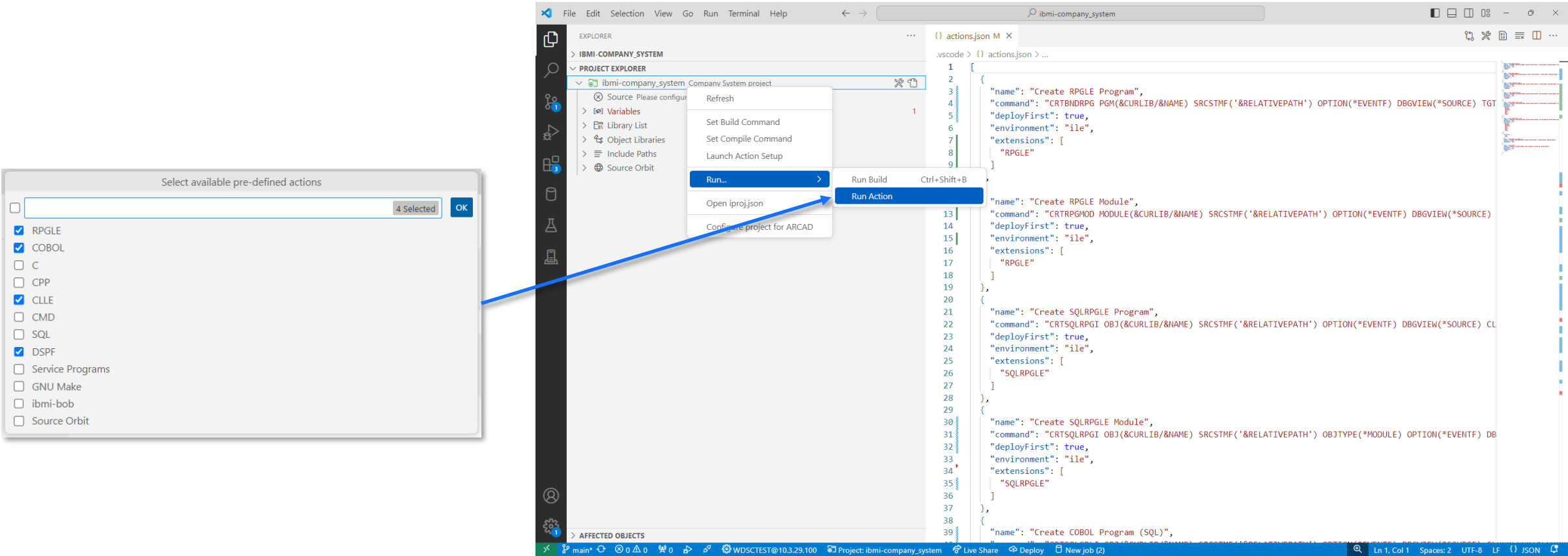
- 1 **Deploy**  
↓
- 2 **Run build or compile command**  
↓
- 3 **Download logs and event files**

- Building
  - Set build command
  - Run Build
- Compiling
  - Set compile command
  - Run compile
    - On active editor
    - On file or directory in File Explorer
    - On file or directory in Source



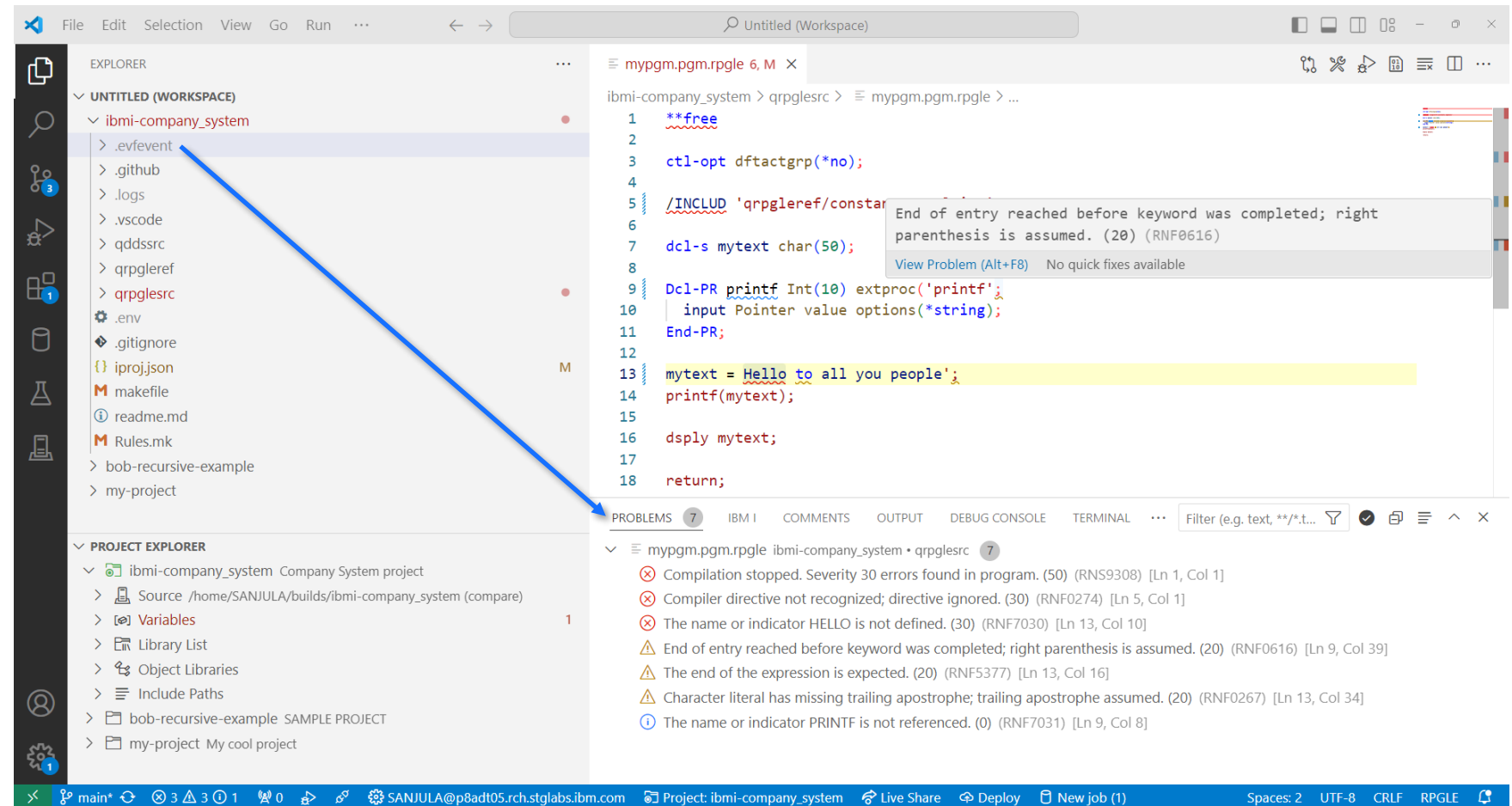
# Run Actions

Project Explorer also still supports running Code for IBM i's custom workspace actions



# View Diagnostics

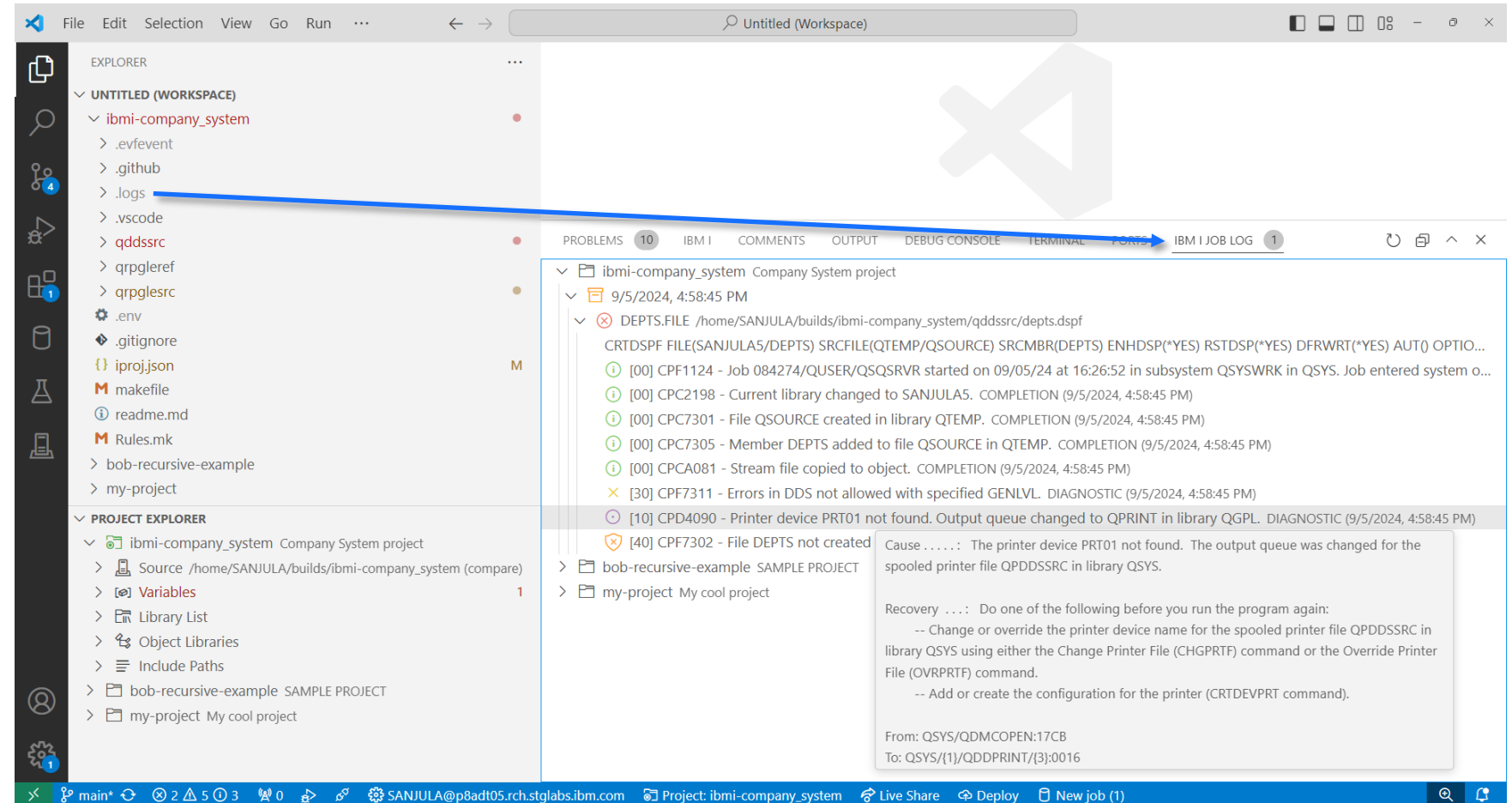
- Evfevent file diagnostics are dumped in .evfevent directory after a build or compile
- Visualize diagnostics in the Problems view
- Diagnostics are also rendered inline in the source file





# View Job Logs

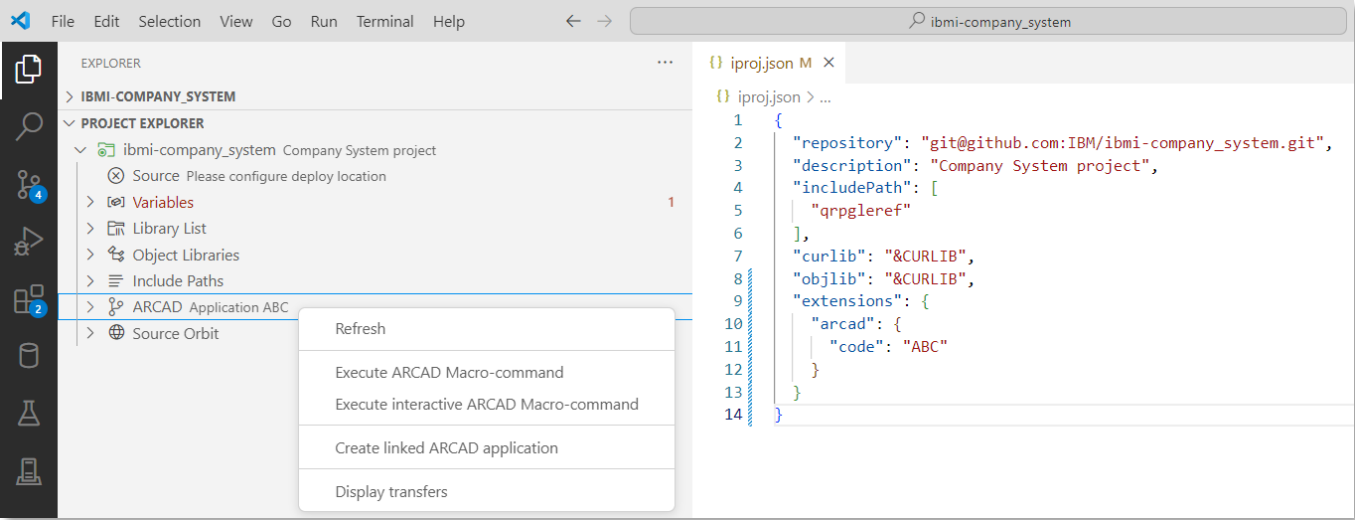
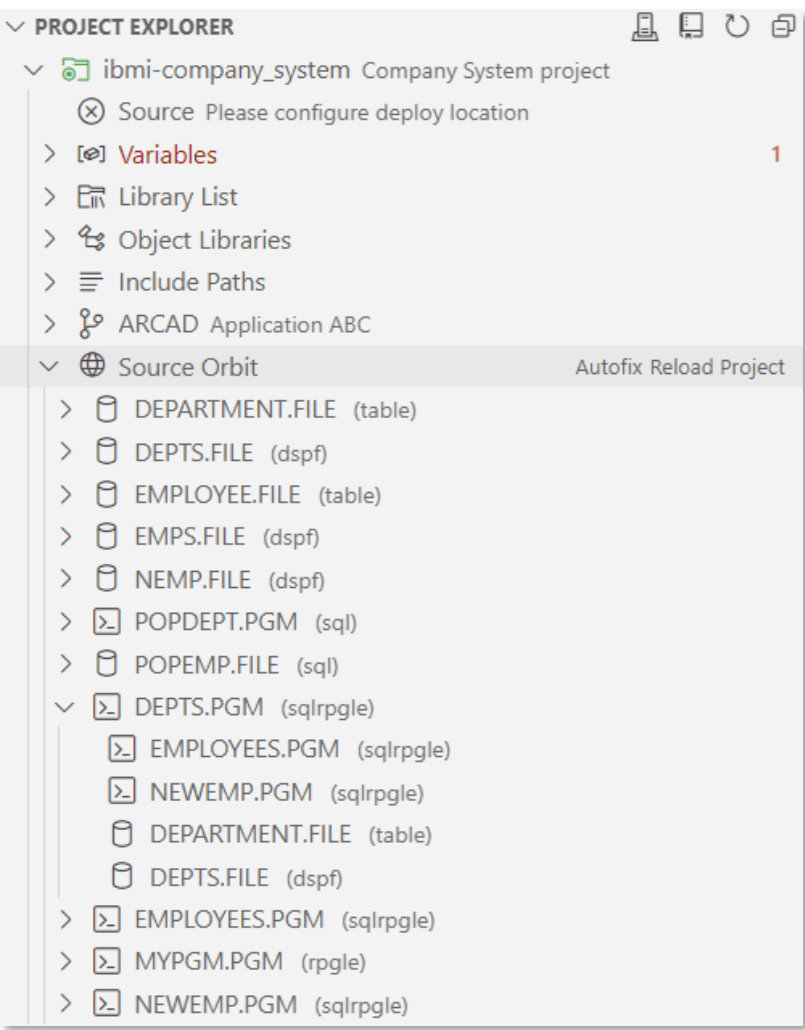
- Job log and spool files are dumped in .logs directory after a build or compile
- Job log view is used to visualize and manage these logs
- Track up to 10 of the previous logs in memory
- Organized by the ILE objects being built
- Filter by failed objects or severity



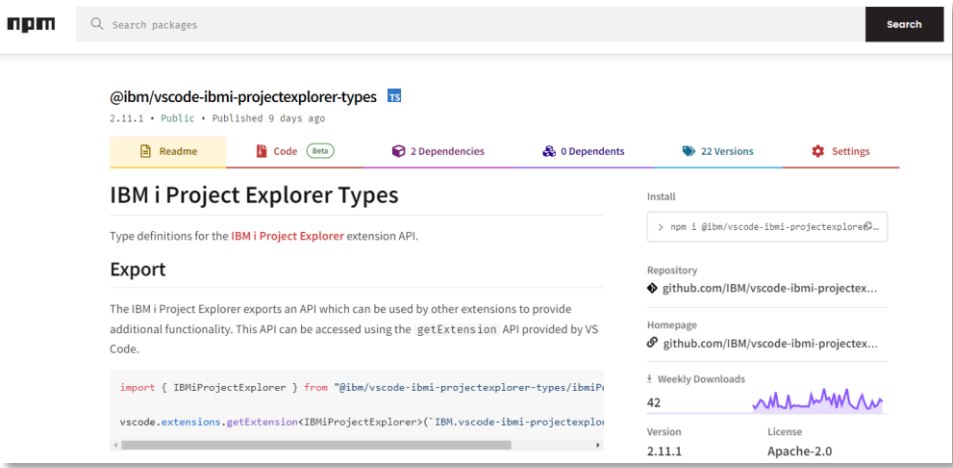
# Integration



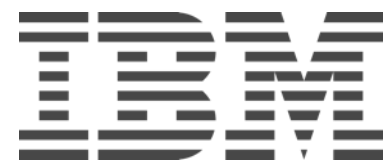
ARCAD-Elias



What can you  
integrate with  
IBM i Project  
Explorer's API?



# Demo



# Links

## IBM i Project Explorer

- VS Code Marketplace <https://marketplace.visualstudio.com/items?itemName=IBM.vscode-ibmi-projectexplorer>
- Documentation <https://ibm.github.io/vscode-ibmi-projectexplorer/#/>
- GitHub Repository <https://github.com/IBM/vscode-ibmi-projectexplorer>
- API <https://www.npmjs.com/package/@ibm/vscode-ibmi-projectexplorer-types>

## Bob

- Documentation <https://ibm.github.io/ibmi-bob/#/>
- GitHub Repository <https://github.com/IBM/ibmi-bob>

## Code for IBM i

- VS Code Marketplace <https://marketplace.visualstudio.com/items?itemName=HalcyonTechLtd.code-for-ibmi>
- Documentation <https://codefori.github.io/docs/#/>
- GitHub Repository <https://github.com/codefori/vscode-ibmi>
- API <https://www.npmjs.com/package/@halcyontech/vscode-ibmi-types>