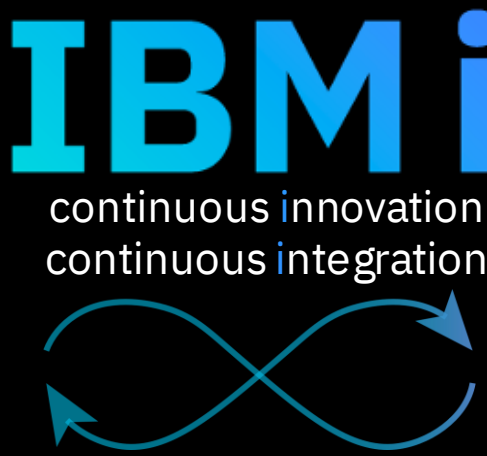
Modern, Buildable Projects
with IBM i Project Explorer and Bob



Edmund Reinhardt
Product Architect - IBM i Application Development
edmund.reinhardt@ca.ibm.com

Sanjula Ganepola
Software Developer
sanjula.ganepola@ibm.com

Agenda

IBM i
continuous innovation
continuous integration

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

IBM i
continuous innovation
continuous integration

Challenges with Building on IBM i

Building on IBM i is hard...

IBM i
continuous innovation
continuous integration

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

IBM i
continuous innovation
continuous integration

MYPROJECT
- QRPGLESRC
  - 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

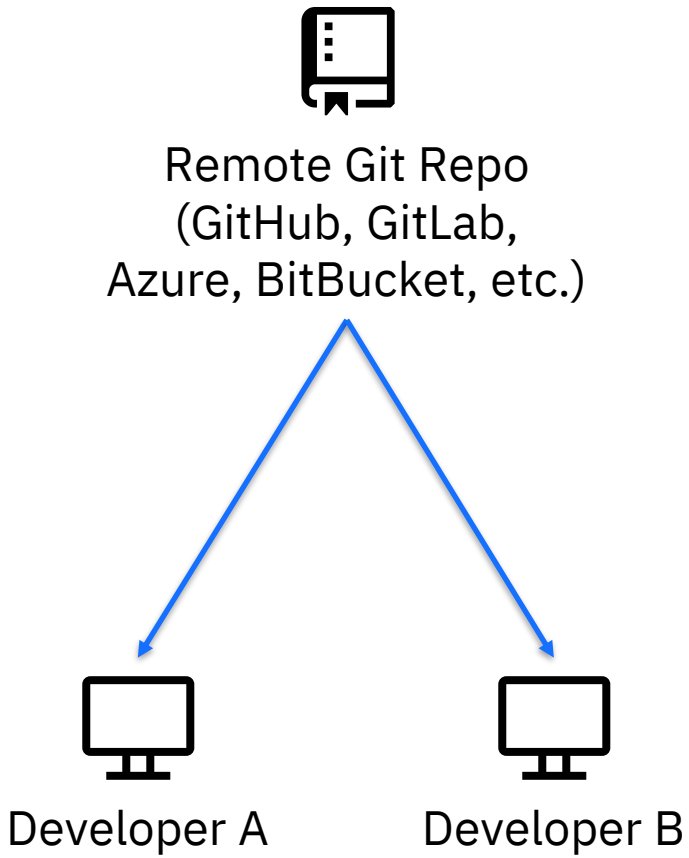
/my-project
- /.git
- qrpglesrc
  - programa.rpgle
  - programb.rpgle
  - programc.rpgle
- qsqlsrc
  - customers.sql
  - inventory.sql
- qcllesrc
  - start.clle
- qcmdsrc
  - Startjob.cmd

IFS/Local File System

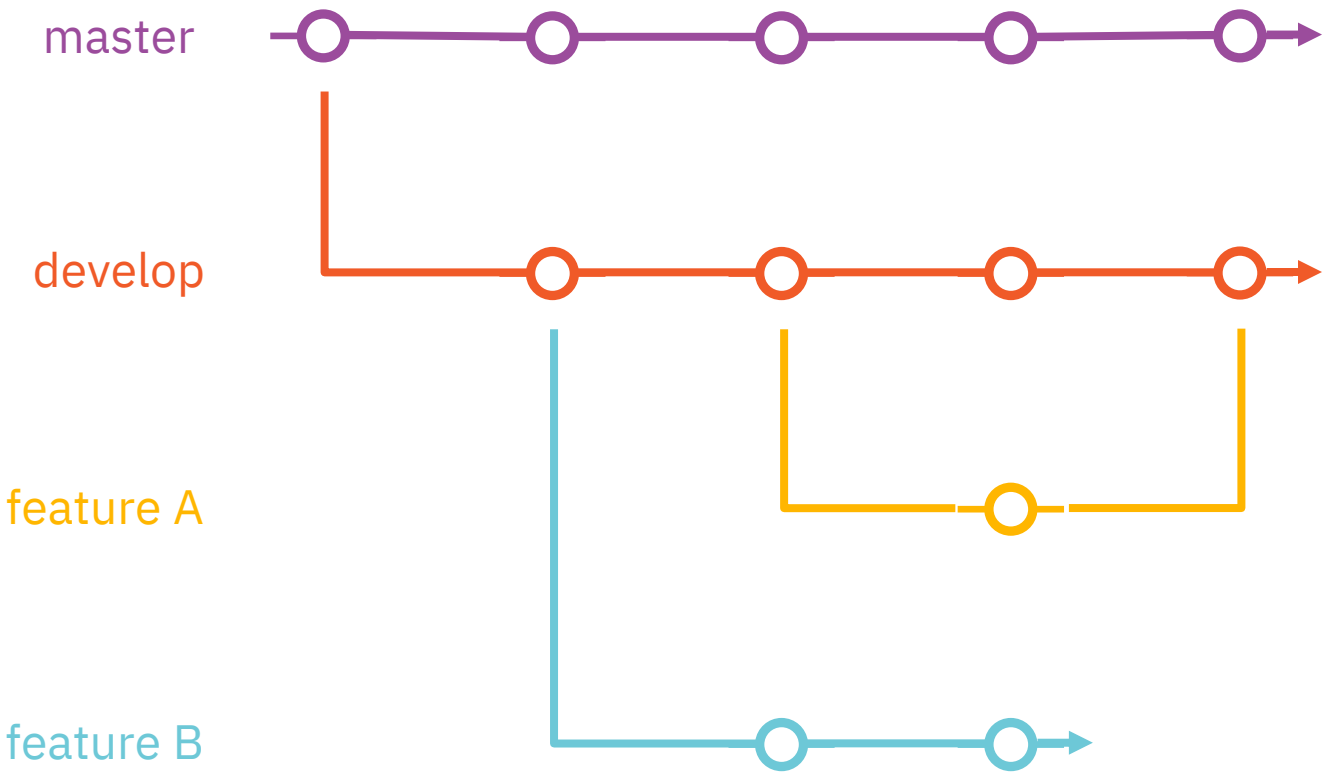
Unlocking source control with Git

IBM i
continuous innovation
continuous integration

Distributed Development



Version Control and Git Workflow



Projects that self-describe how to build themselves!?

IBM i
continuous innovation
continuous integration

- my-project
  - .evfevent
    - file.evfevent
    - source.evfevent
  - .logs
    - joblog.json
    - file.splf
    - source.splf
  - includes
    - constants.rpgleinc
  - QRPGLESRC
    - .ibmi.json
    - Rules.mk
    - source.rpgle
  - QDDSSRC
    - Rules.mk
    - file.pf
  - .env
  - .gitignore
  - .ibmi.json
  - iproj.json
  - Rules.mk

Project Information

Configure library list

Configure build/compile environment

iproj.json

```
{
    "version": "0.0.1",
    "description": "SAMPLE PROJECT",
    "repository": "https://github.com/edmundreinhardt/bob-recursive-example.git",
    "license": "Apache 2.0",
    "objlib": "&CURLIB",
    "curlib": "&CURLIB",
    "includePath": [
        "includes",
        "QPROTOSRC"
    ],
    "preUsrlibl": [
        "&lib1"
    ],
    "postUsrlibl": [
        "&lib2"
    ],
    "setIBMiEnvCmd": [],
    "compileCommand": "makei c -f {filename}",
    "buildCommand": "makei build"
}
```

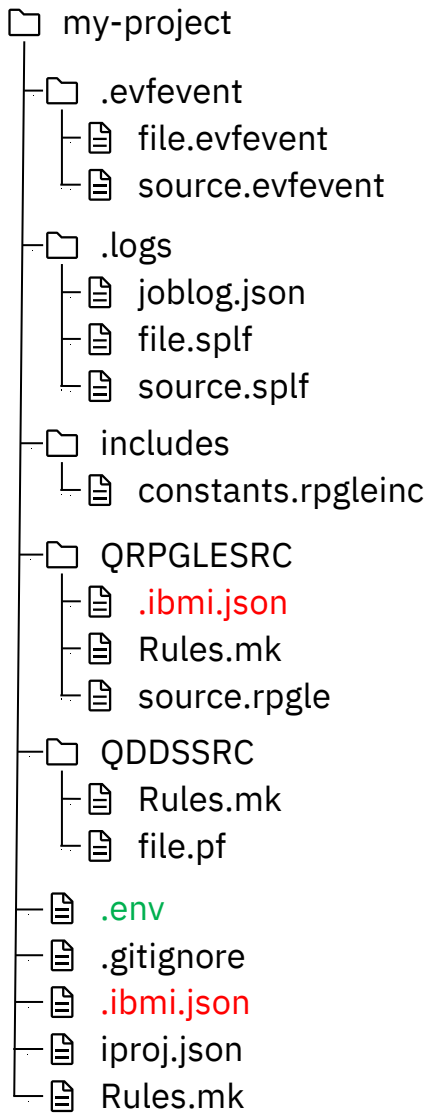
Standardized metadata format with variables (&...)

Set build/compile command

iproj.json in project root

Flexible subdirectories and build customization

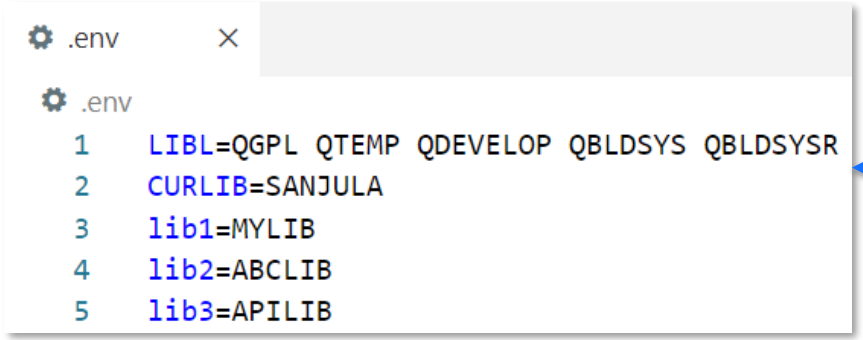
IBM i
continuous innovation
continuous integration



Target object library for directory

EBCDIC encoding for compiler

.ibmi.json in subdirectories



Custom variable values so that each developer can customize build

.env in project root

Control what objects to build and how to build them

IBM i
continuous innovation
continuous integration

- my-project
  - .evfevent
    - file.evfevent
    - source.evfevent
  - .logs
    - joblog.json
    - file.splf
    - source.splf
  - includes
    - constants.rpgleinc
  - QRPGLESRC
    - .ibmi.json
    - Rules.mk
    - source.rpgle
  - QDDSSRC
    - Rules.mk
    - file.pf
  - .env
  - .gitignore
  - .ibmi.json
  - iproj.json
  - Rules.mk

Rules.mk

```
1   SUBDIRS = qrpglesrc qddssrc
```

Declare subdirectories to be built

Rules.mk in project root

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
```

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

Rules.mk in subdirectories

Build and Compile Process

IBM i
continuous innovation
continuous integration

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

IBM i
continuous innovation
continuous integration

Ins and Outs of IBM i Project Explorer

Overview

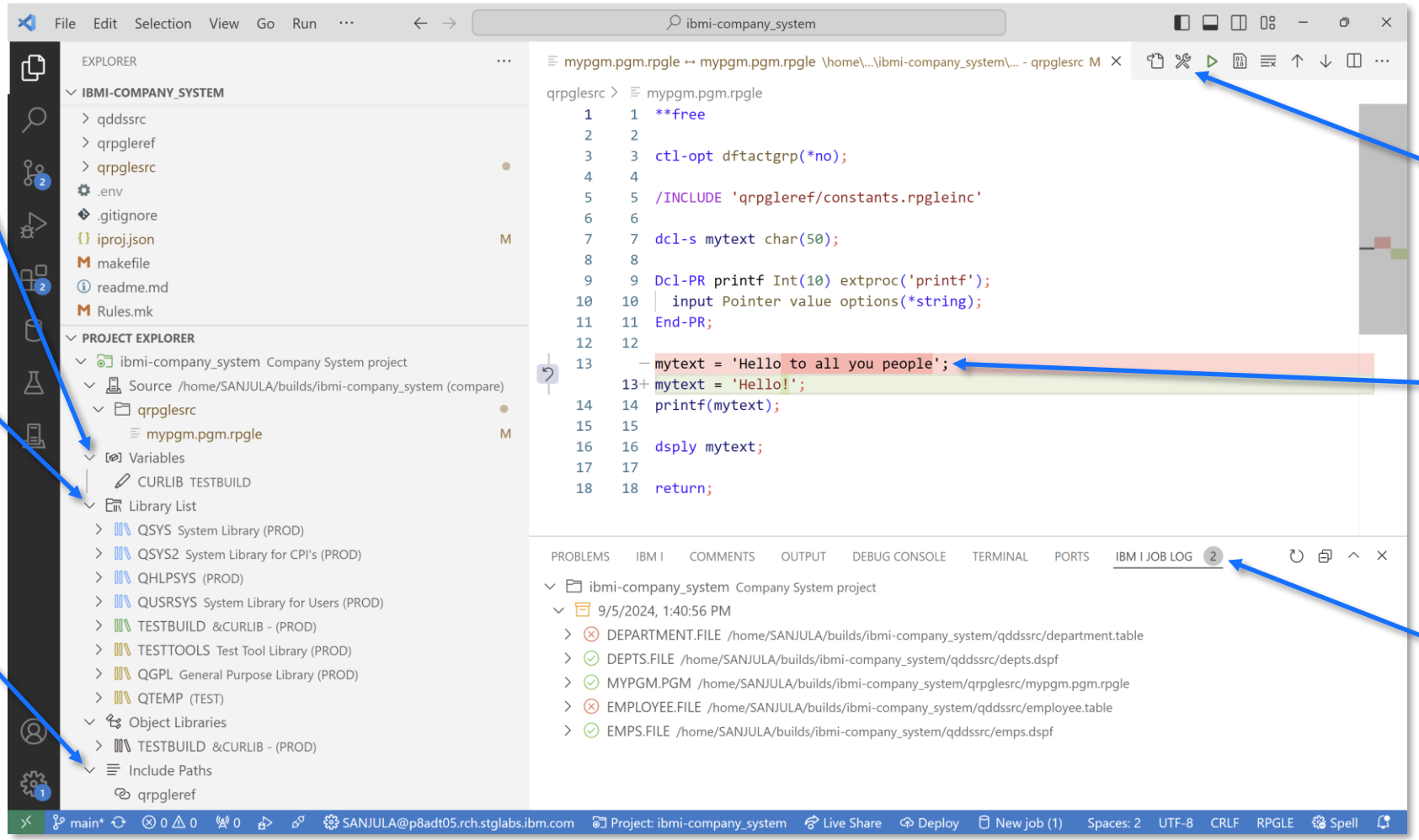
IBM i
continuous innovation
continuous integration

The ultimate tool for local development on IBM i!

Set variables

Manage library list

Modify include paths



Build and Compile

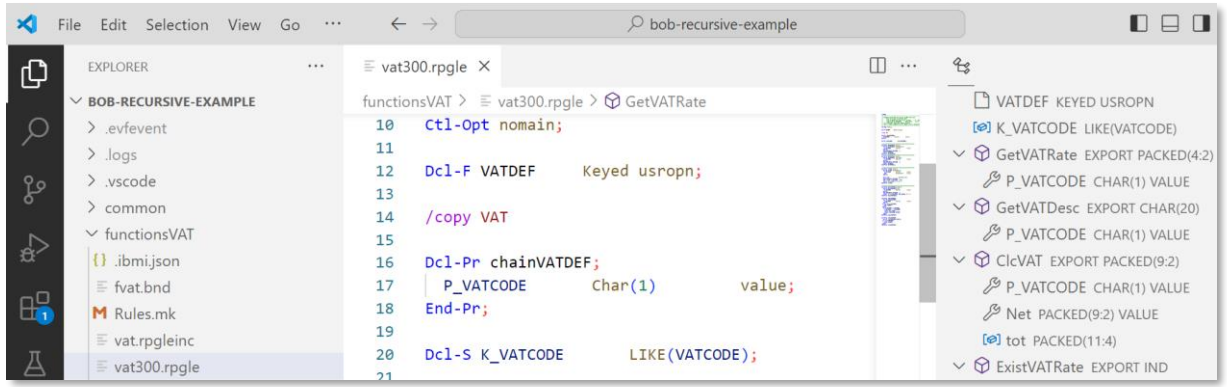
Local source vs. IFS source

View job logs

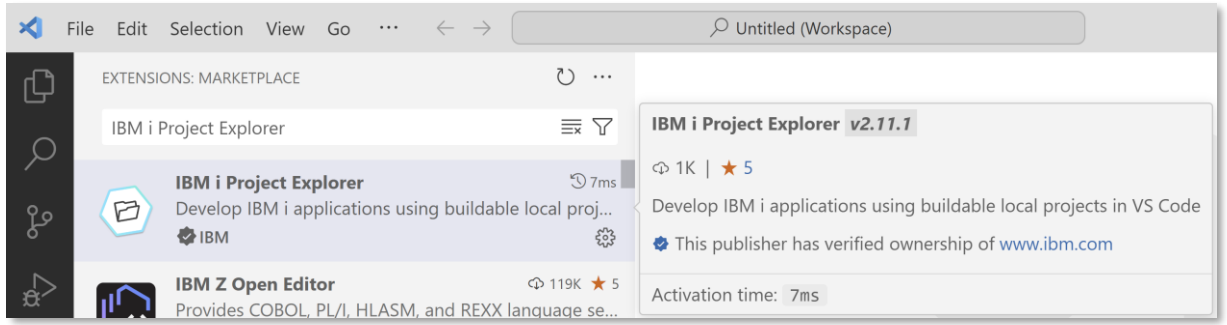
Installation

IBM i
continuous innovation
continuous integration

1 *Download*
Visual Studio Code



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

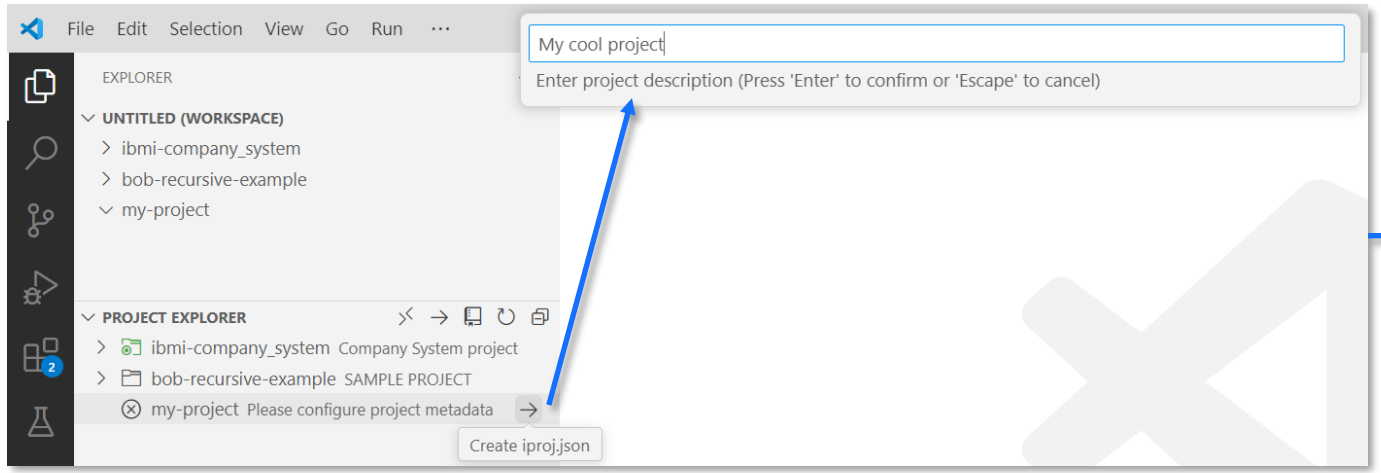


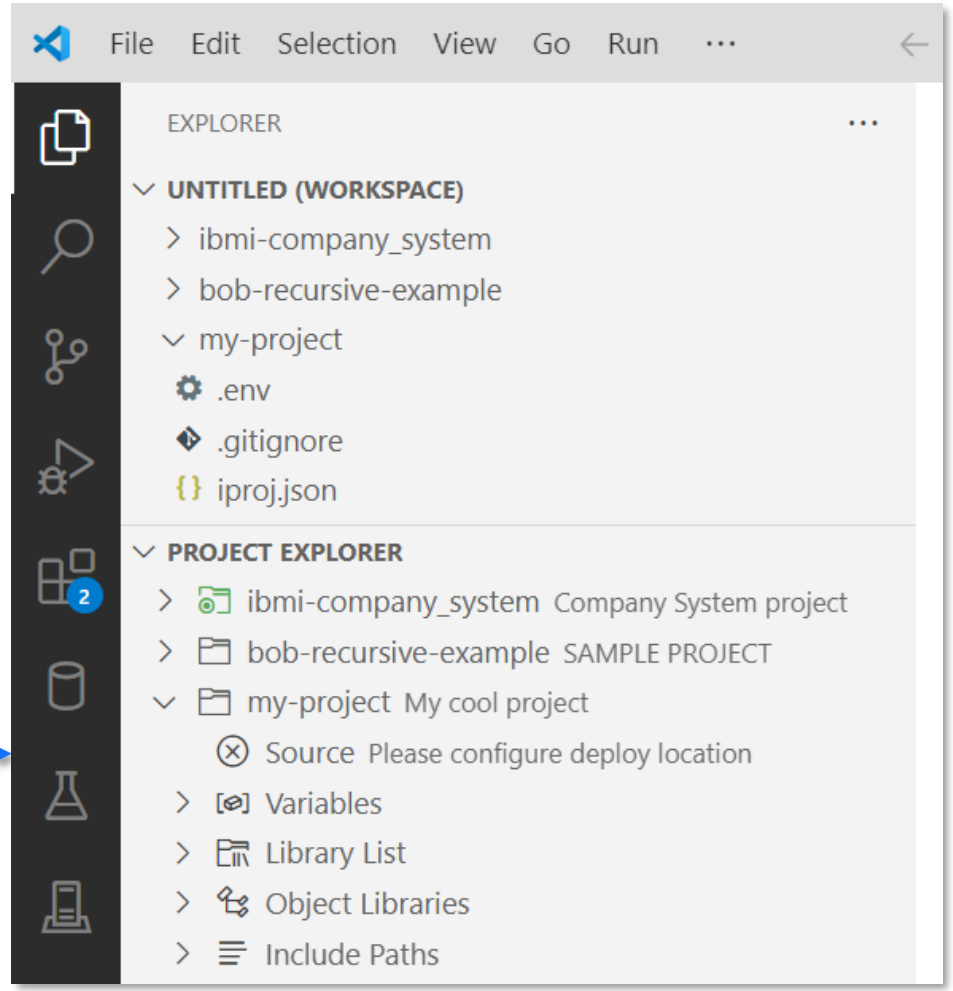
3 *Run*
yum install bob
on IBM i

Create a New Project

IBM i
continuous innovation
continuous integration

- 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

IBM i
continuous innovation
continuous integration

CVTSRCPF from BOB

QSYS members in source physical files

↓

Properly encoded, terminated, and named source files in an IFS directory

↓

Download to local project

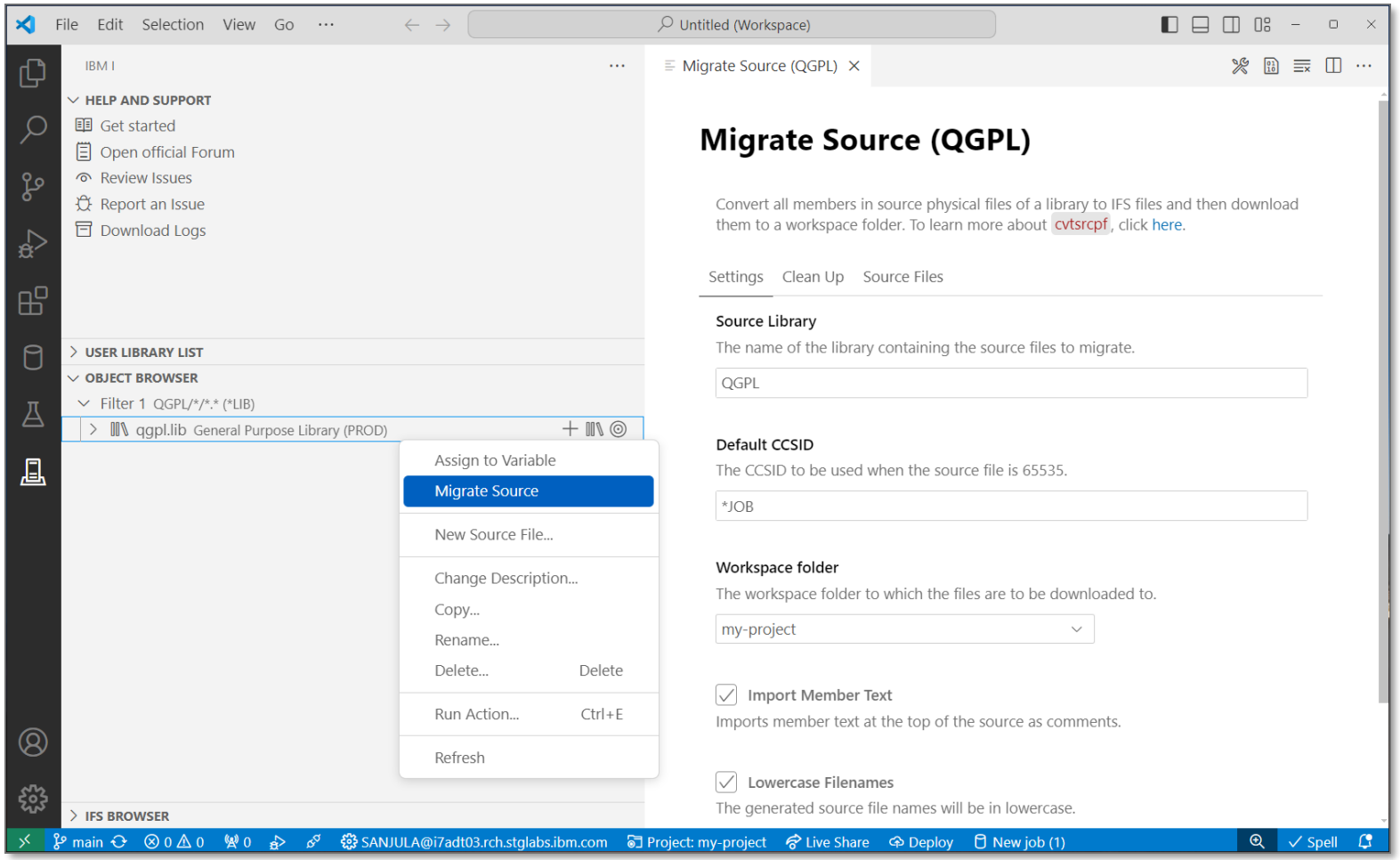
↓

Source Orbit

Rename extensions

↓

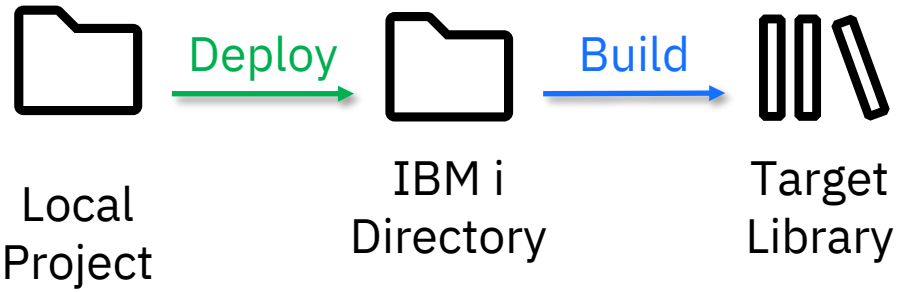
Convert includes/copy directives to Unix style paths

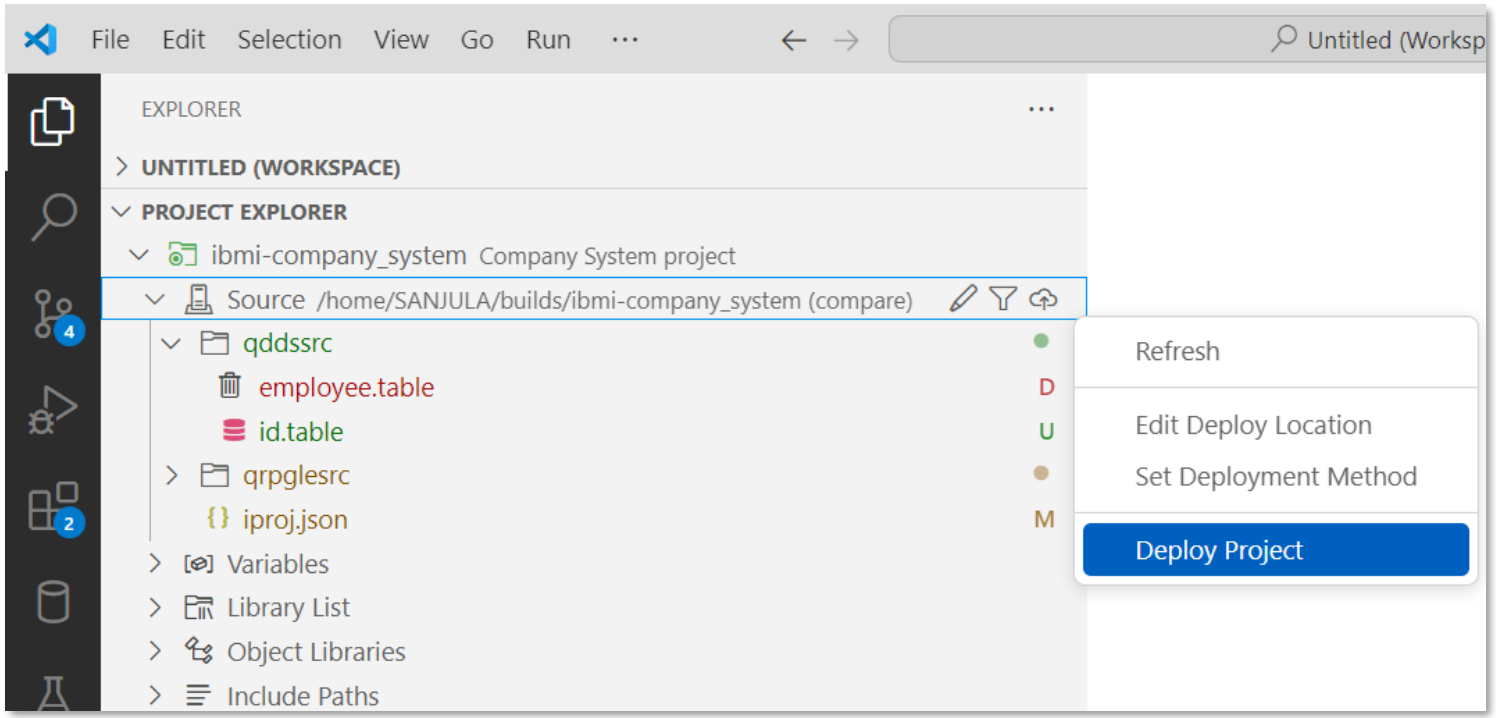


Source and Deployment

IBM i
continuous innovation
continuous integration

- 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



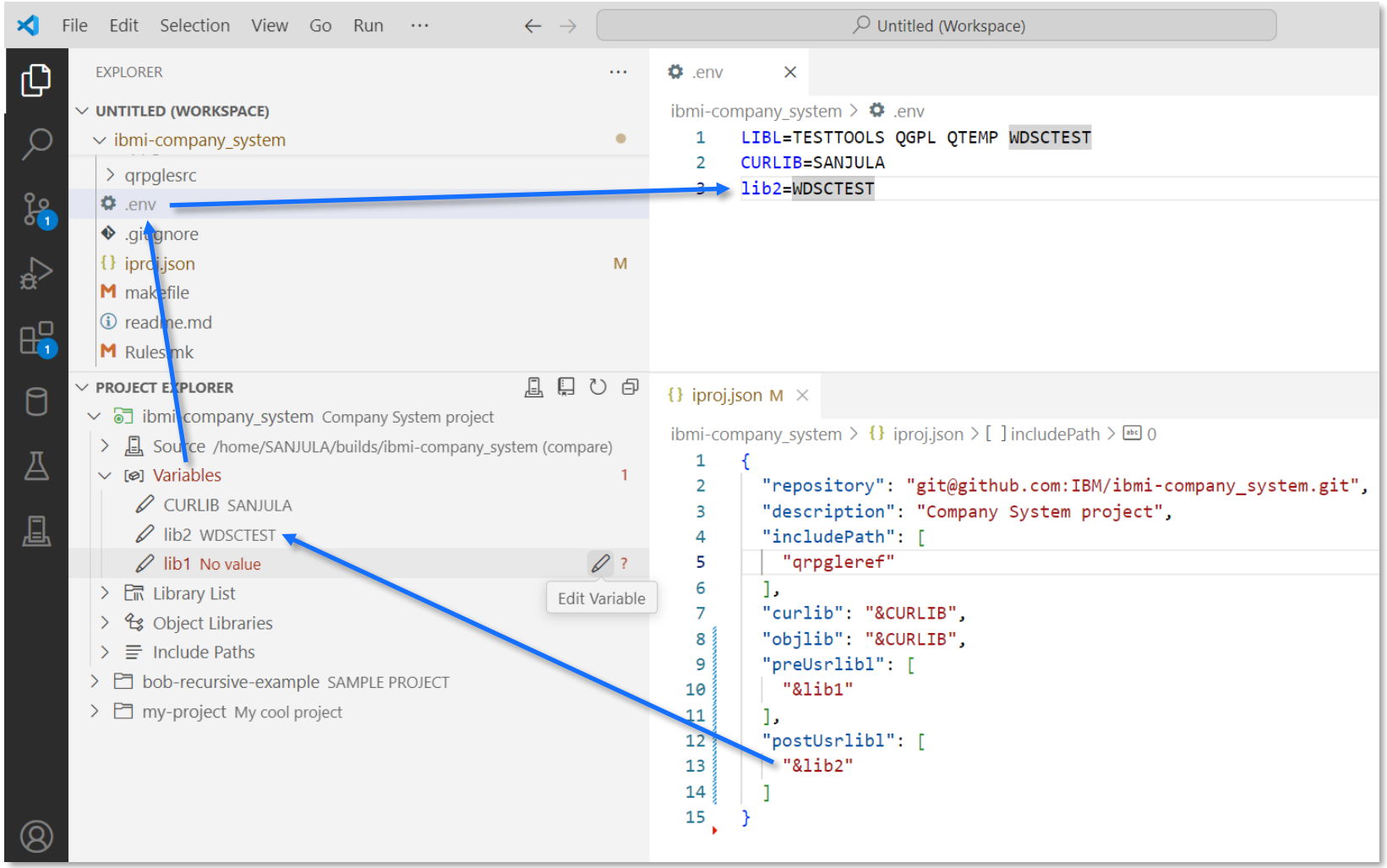


Work with Variables

IBM i
continuous innovation
continuous integration

- 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

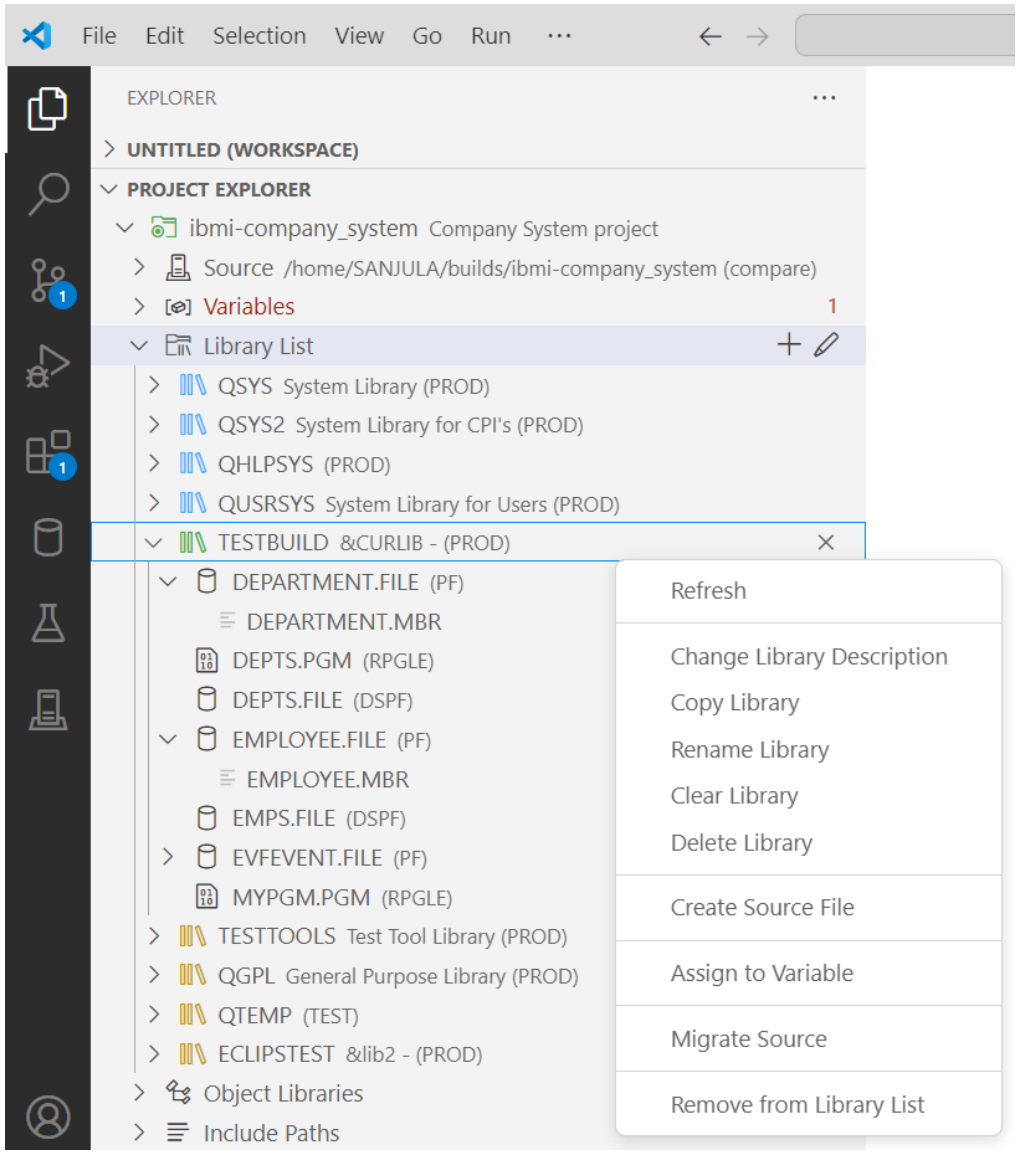
Do not push .env file to Git!



Manage the Library List

IBM i
continuous innovation
continuous integration

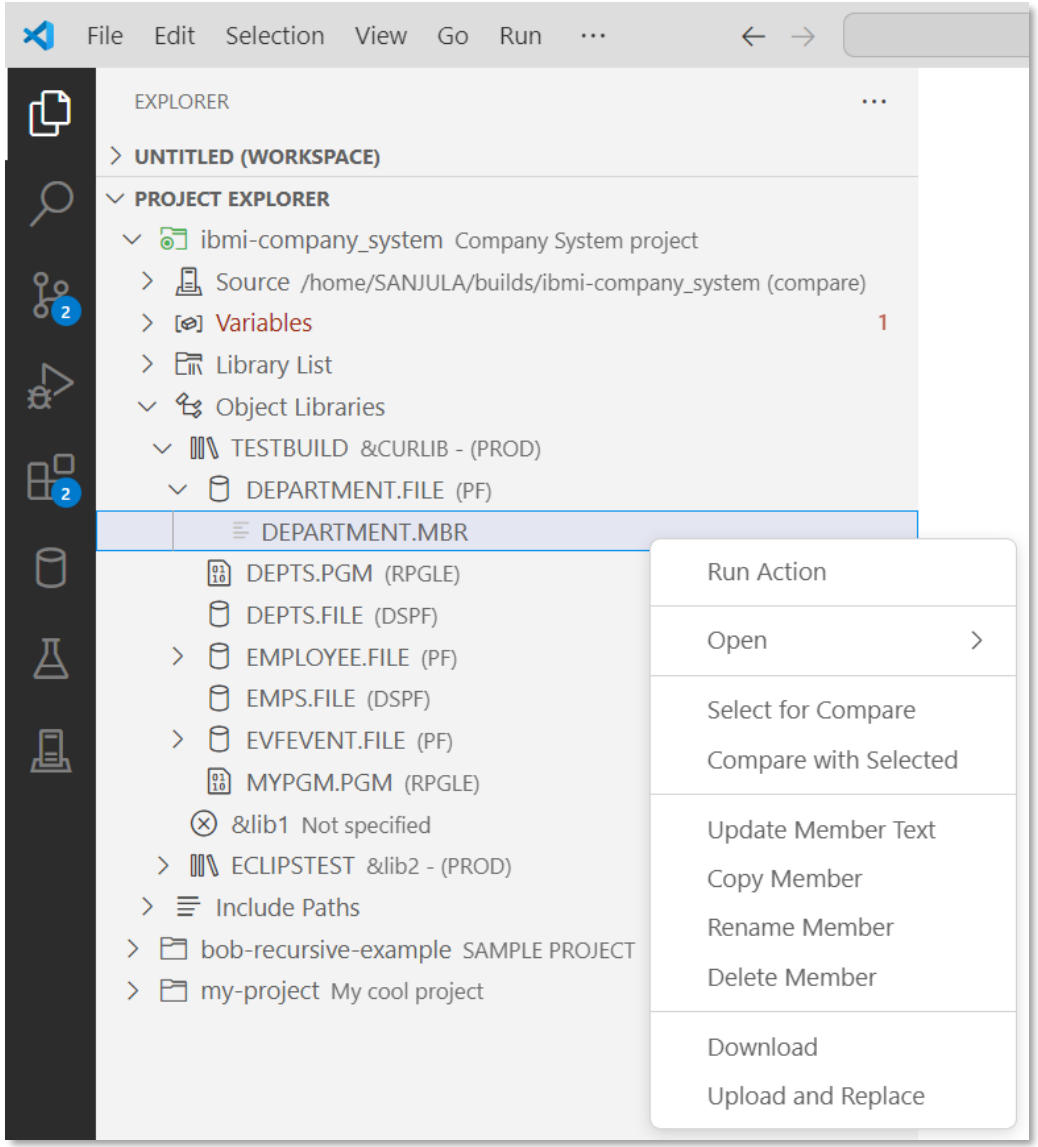
- Add to beginning/end of library list (preUsrlibl and postUsrlibl) and set current library (curlib in iproj.json)
- Reorder library list
- Browse objects and members
- Manage libraries, objects, and members



Browse Object Libraries

IBM i
continuous innovation
continuous integration

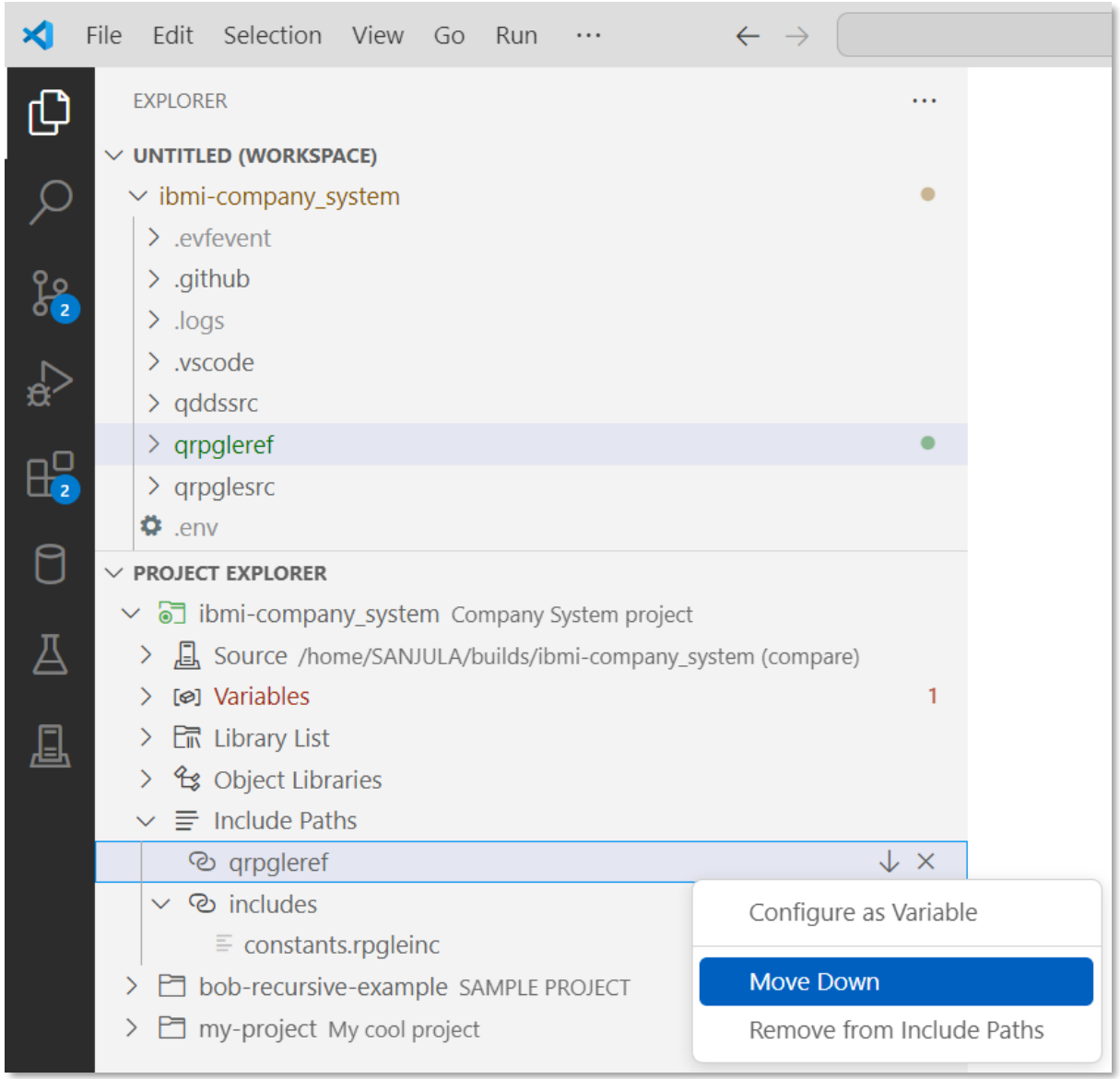
- Another place to manage libraries in iproj.json (curlib, objlib, preUsrlibl, postUsrLibl)
- Manage libraries, objects, and members



Manage Include Paths

IBM i
continuous innovation
continuous integration

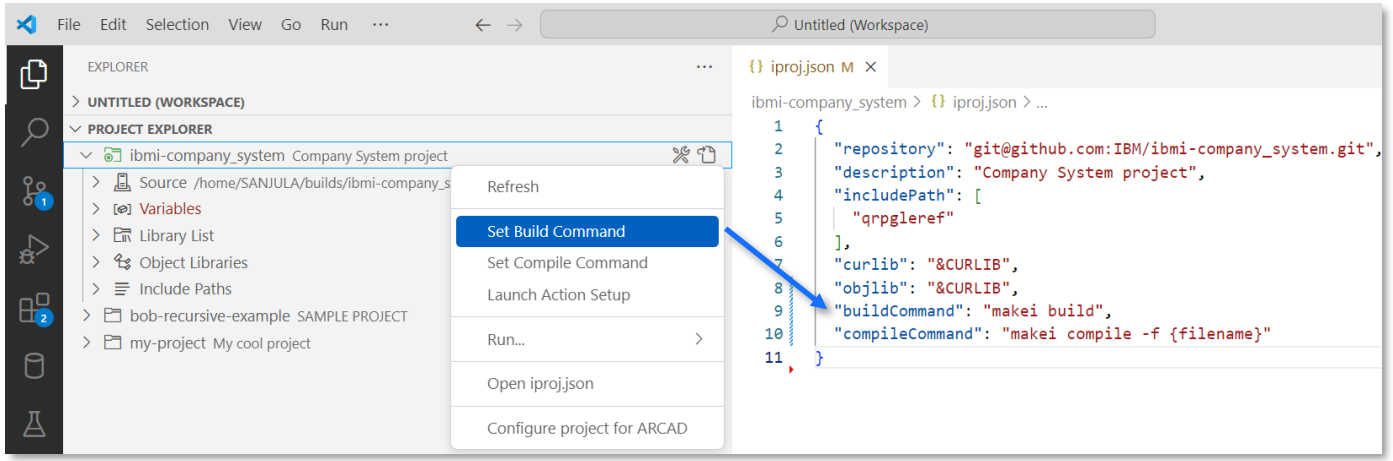
- Add, remove, and reorder include paths
- Visualize if includes resolve locally or to remote IFS



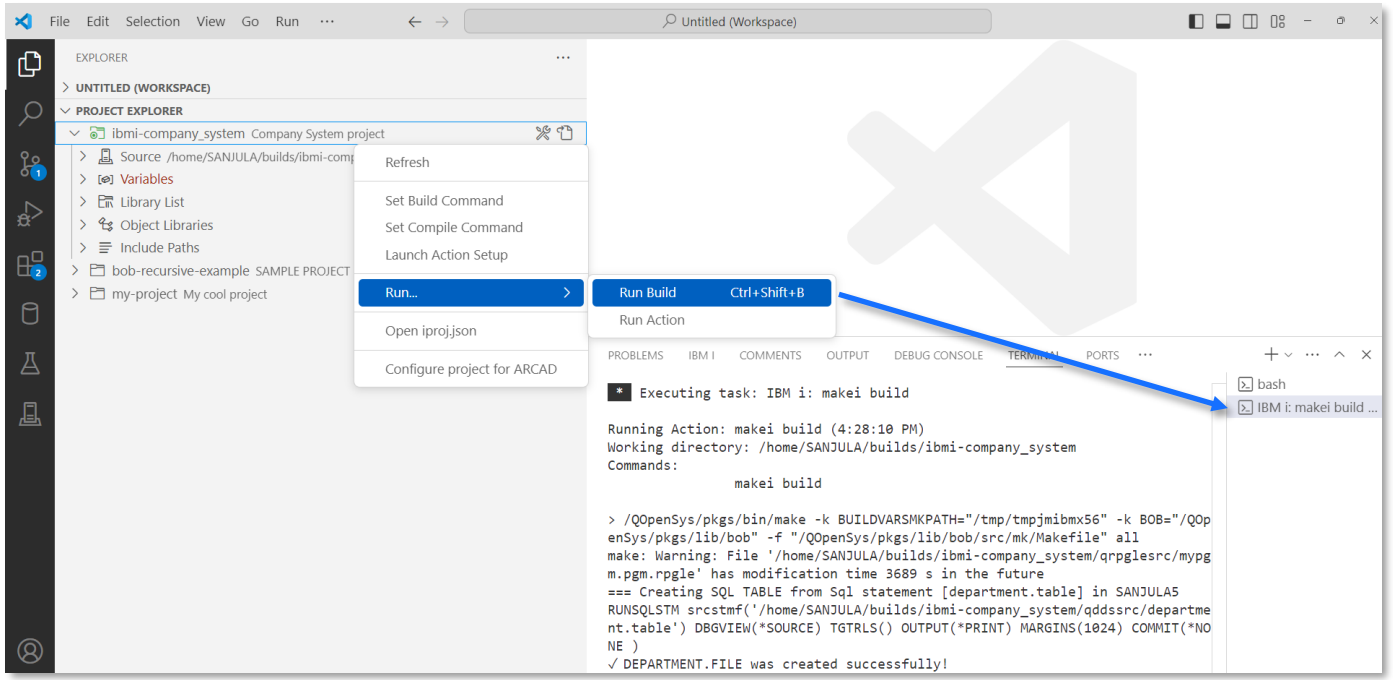
Build and Compile

IBM i
continuous innovation
continuous integration

1. Deploy
↓
2. Run build or compile command
↓
3. Download logs and evfevent files



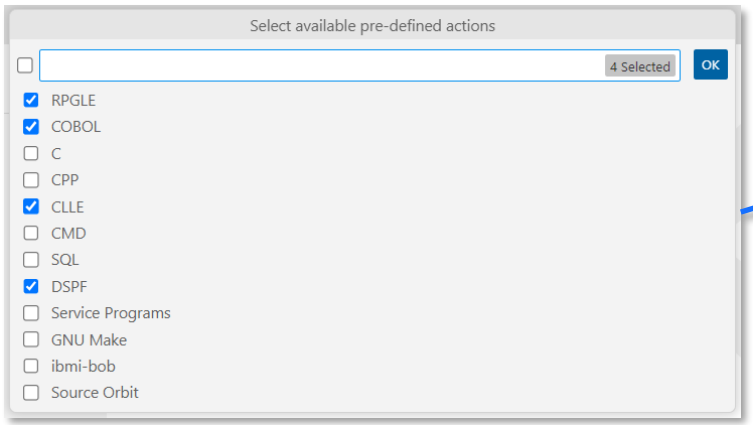
- Building
  - Set build command
  - Run Build
- Compiling
  - Set compile command
  - Run compile
    - On active editor
    - On file or directory in File Explorer
    - On file or directory in Source

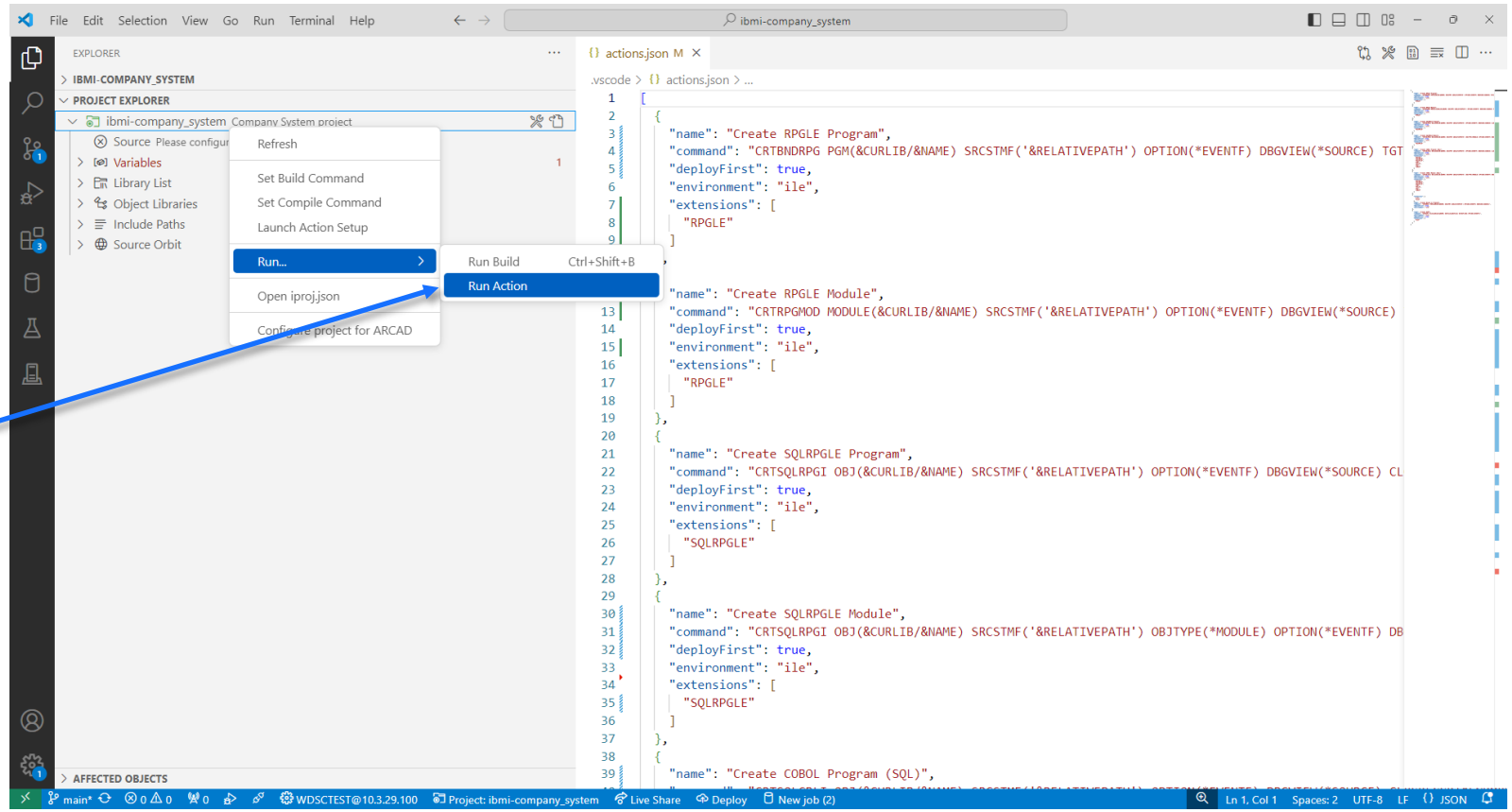


Run Actions

IBM i
continuous innovation
continuous integration

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

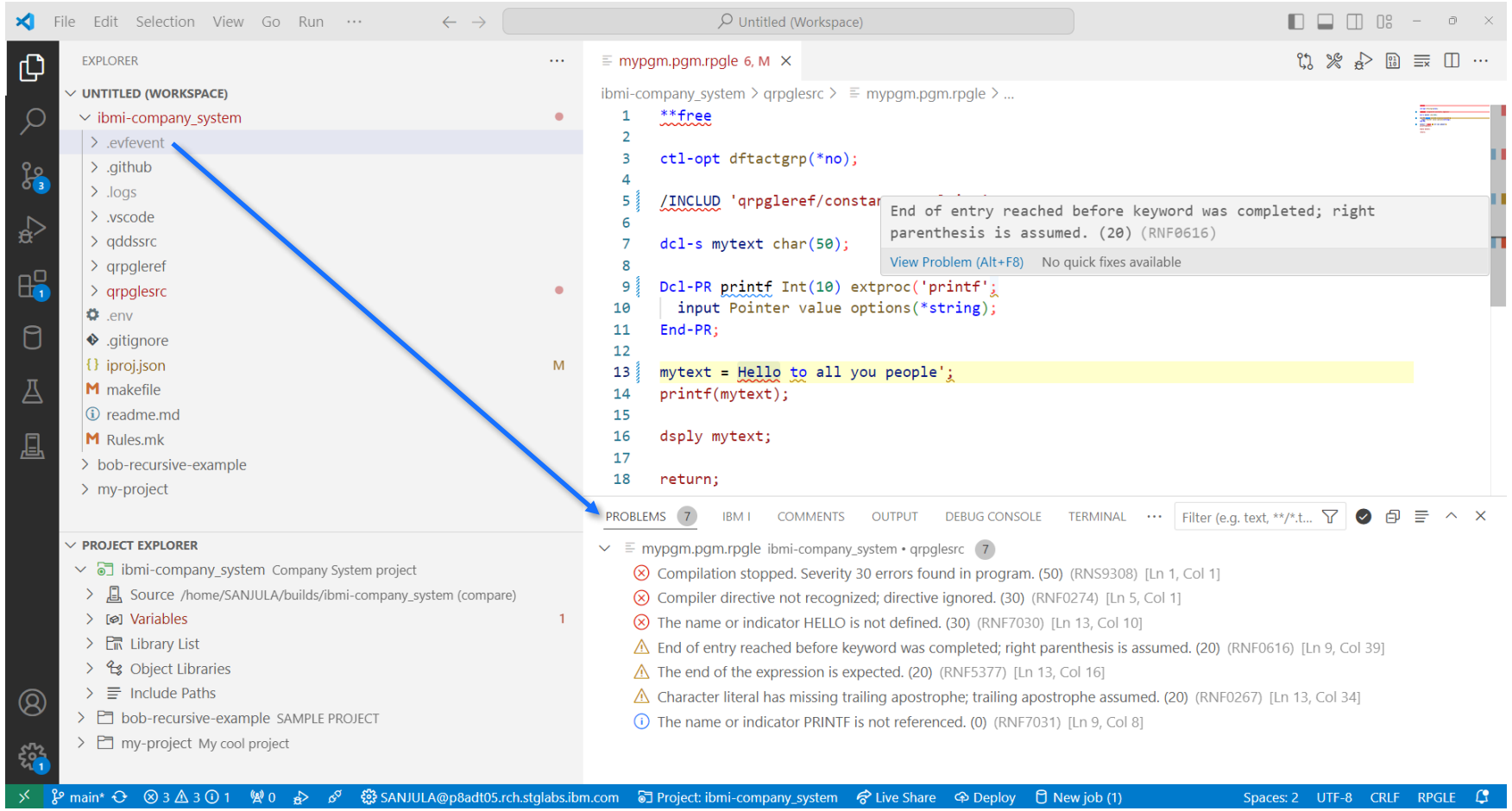




View Diagnostics

IBM i
continuous innovation
continuous integration

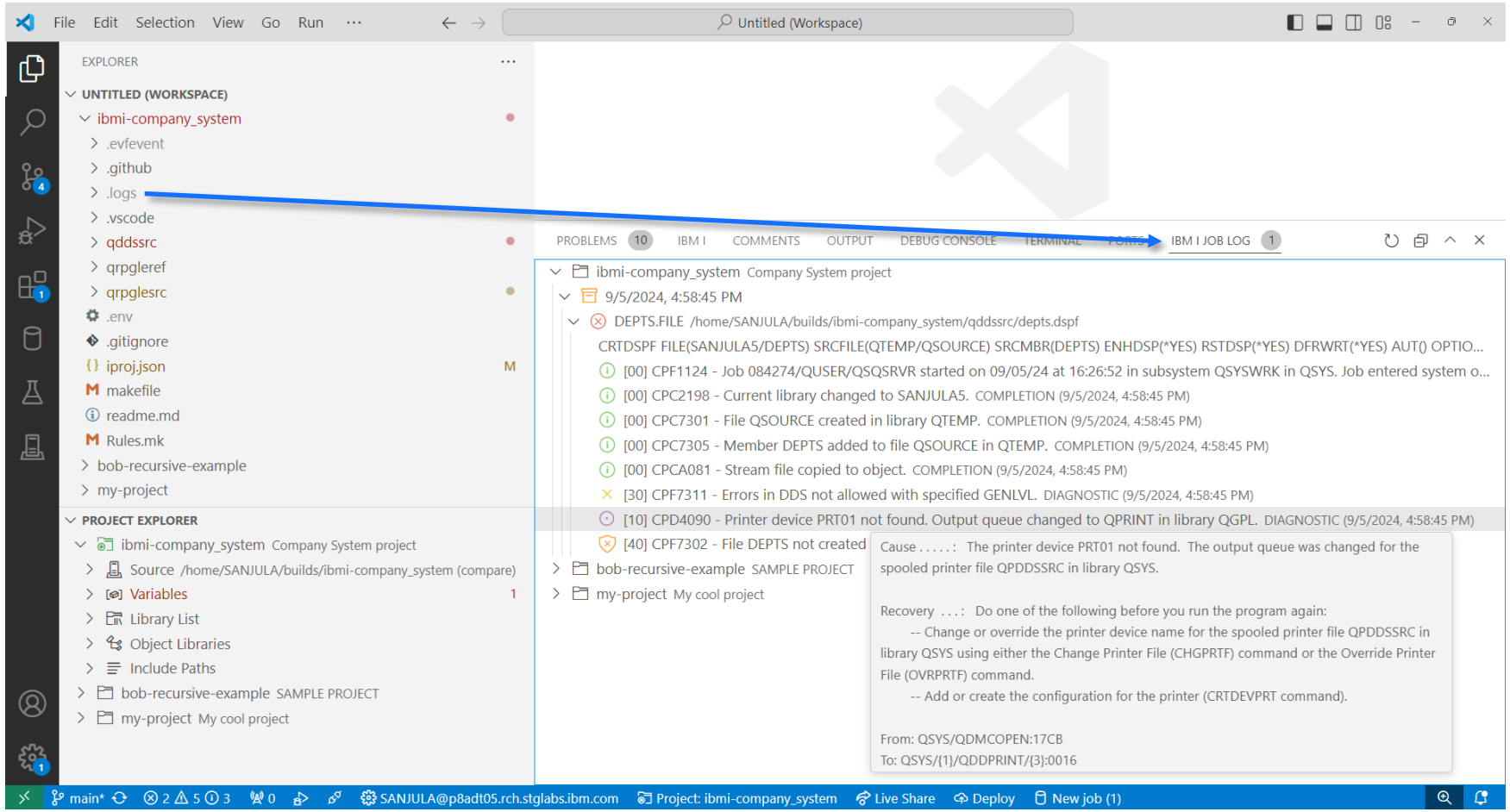
- Evfevent file diagnostics are dumped in .evfevent directory after a build or compile
- Visualize diagnostics in the Problems view
- Diagnostics are also rendered inline in the source file



View Job Logs

IBM i
continuous innovation
continuous integration

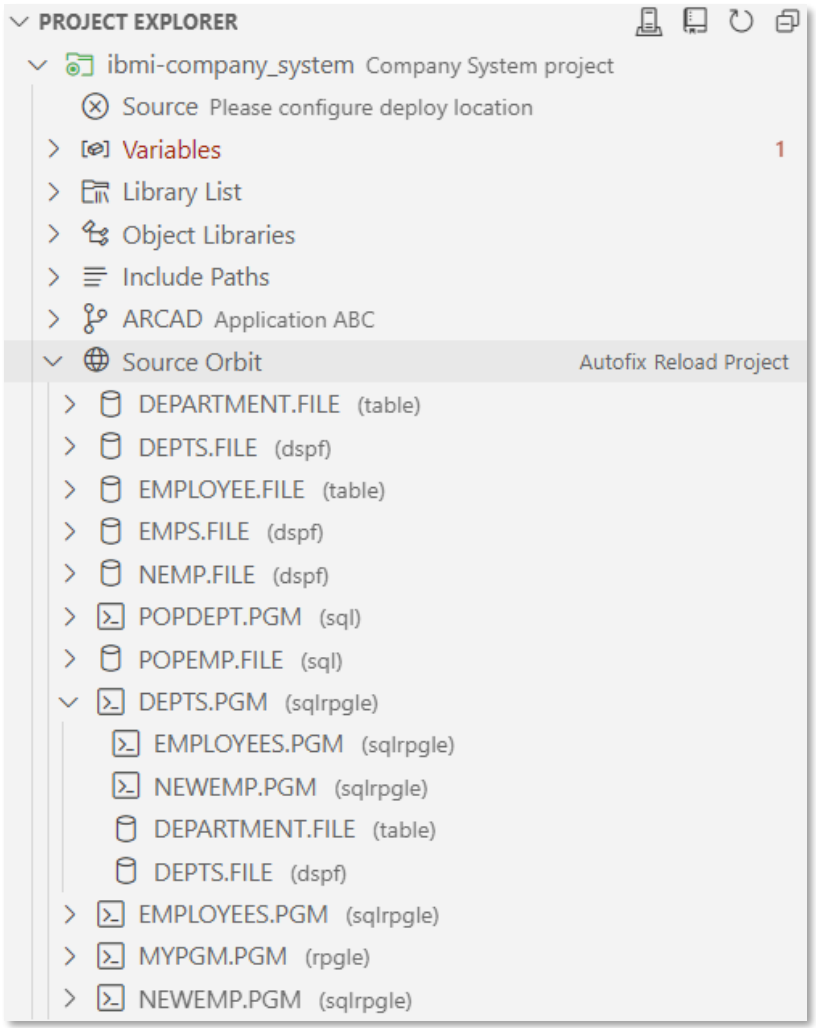
- Job log and spool files are dumped in .logs directory after a build or compile
- Job log view is used to visualize and manage these logs
- Track up to 10 of the previous logs in memory
- Organized by the ILE objects being built
- Filter by failed objects or severity



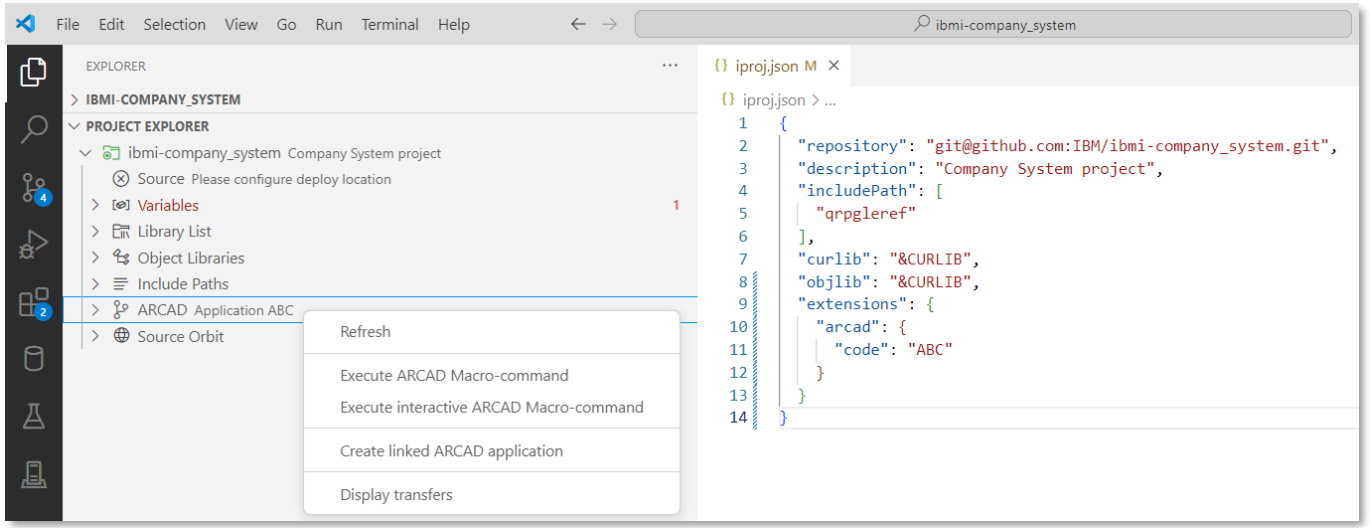
Integration

IBM i
continuous innovation
continuous integration

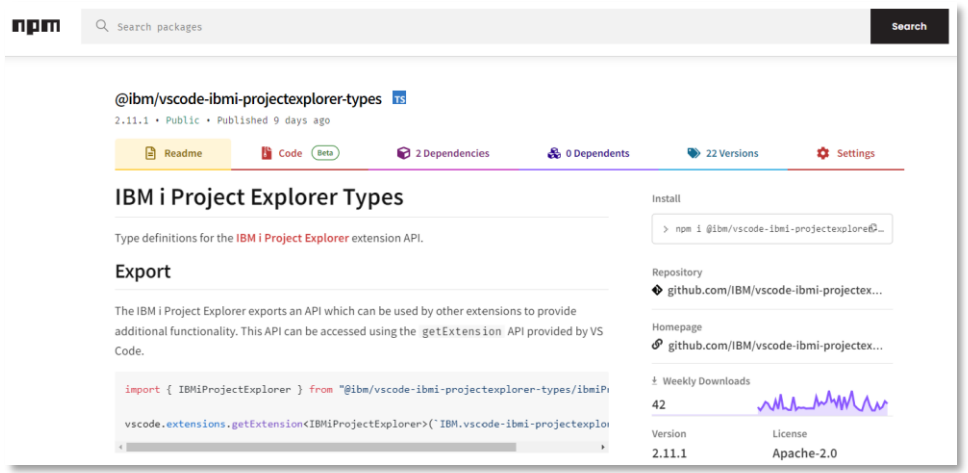
Source Orbit



ARCAD-Elias



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



Demo

Links

IBM i
continuous innovation
continuous integration

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