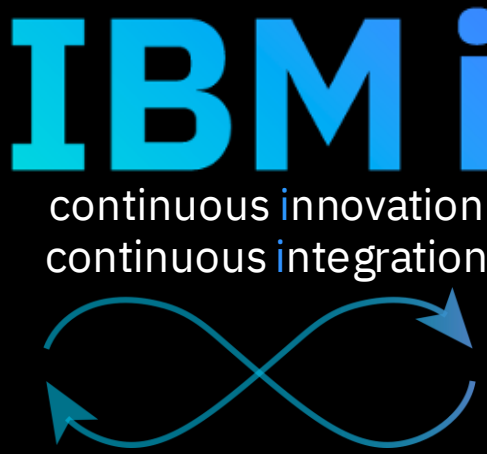
Modern, Buildable Projects
with IBM i Project Explorer and Bob



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
- Bob (Better Object Builder) for IBM I
- Local Development
- IBM i Project Explorer
- Demo

IBM i
continuous innovation
continuous integration

Challenges with Building on IBM i

General Challenges

IBM i
continuous innovation
continuous integration

- 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)

- Libraries
  - Only 2 level hierarchy to organize, with only short 10 char names

- Source control
  - None (sequence number dates)
  - Home grown
  - Proprietary IBM i systems
    - Cost
    - Smaller market = less investment

- Build system
  - Individual CRTXXXMOD + CRTPGM
  - CL Scripts
  - A couple of vendors have dependency-based build

RDi Projects – Lessons Learned

IBM i
continuous innovation
continuous integration

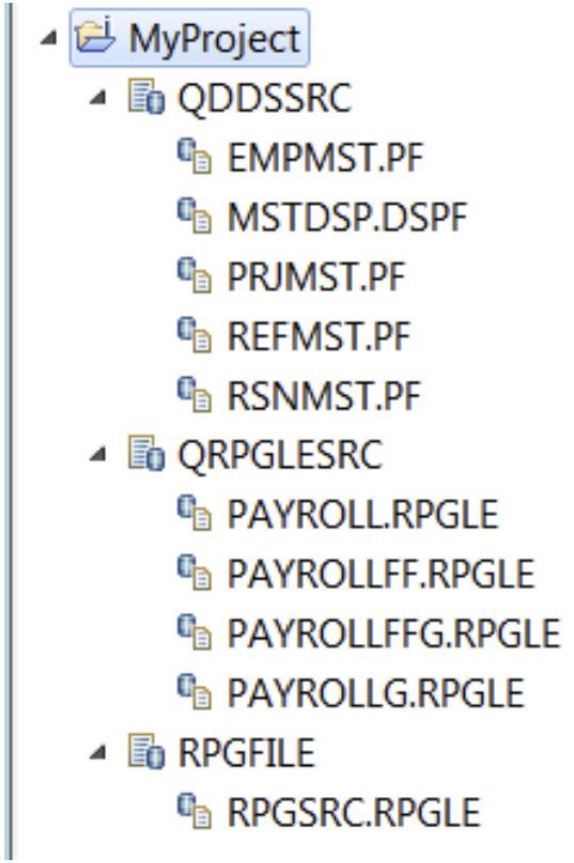
- Supports git but …

- Mapping from i Project to exactly one library was too inflexible
  - Some customers target many libraries from one project (program / data / source)
  - Other customers have huge libraries

- Metadata was very hard to maintain
  - Having a parallel directory hierarchy under .ibmi meant that any time a SRC-PF or member changed names the metadata was lost

- Mapping rigidly to SRC-PF meant inheriting all its limitations
  - Fixed line length
  - Fixed directory hierarchy of basically 1 level with only 10 characters

- Build was very limited
  - No disambiguating of PGM vs MOD
  - No understanding of binding relationships
  - No incremental ability (i.e. only build what had changed)

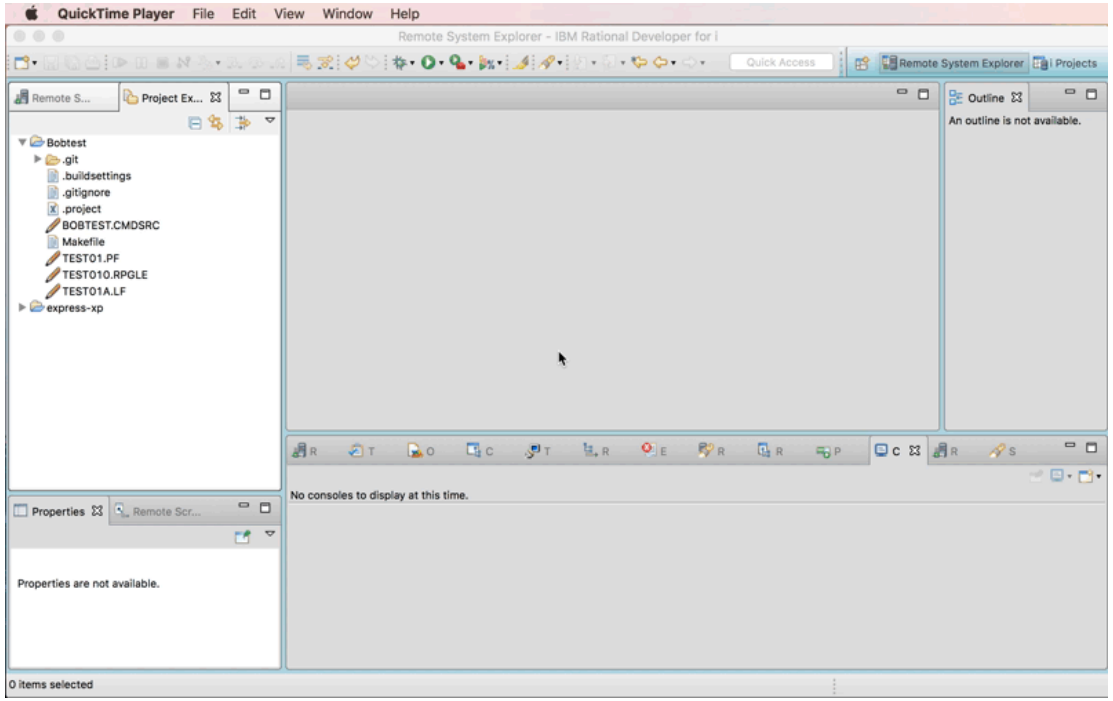


Existing Bob By S4i

IBM i
continuous innovation
continuous integration

- Open-source project by Jeff Berman (https://github.com/s4isystems/Bob)
  - Incremental compile ability based on gmake
  - Some level of ILE binding understanding
  - Not bound to library and SRC-PF structure/naming
  - Member level specific metadata using gmake variable
  - Consideration of target EBCDIC CCSID for compiler
  - Support of old languages whose compilers do not have IFS support yet (DDS, UIM)
  - Retrieval of all EVFEVENT files to enable compiler feedback

- Limitations
  - Uppercase names required
  - Single target library
  - Single directory containing source
  - No metadata on environment prerequisites (i.e. LIBL, ASP, where to find includes, etc.)
  - Install was complex – not yum-enabled
  - No 1 to 1 mapping of file extensions to compile (i.e. are we targeting MOD or PGM)



IBM i
continuous innovation
continuous integration

Bob (Better Object Builder) for IBM i

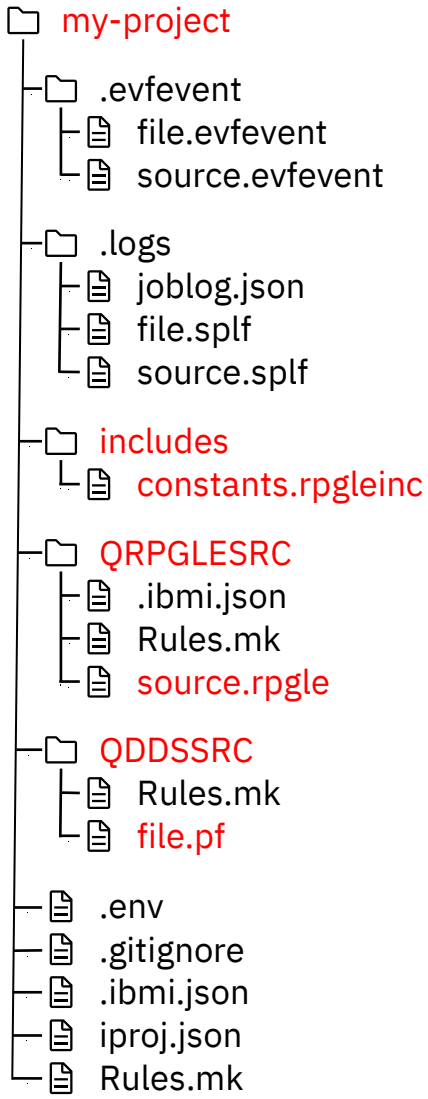
IBM i Bob

IBM i
continuous innovation
continuous integration

- Enhancements
  - Project definition
    - Know how to build yourself
    - Know where to resolve includes
    - Know how to set up environment
    - Still flexible so that what is stored in Git does not have to be modified for each developer or deployment scenario
  - No limit on number of directories and their nesting
  - No limit on directory naming
  - No limit on number of object libraries
  - Unambiguous mapping from file name to compile type

- Usability
  - PASE command line
  - Windows/Mac command line with rsync/scp to do file transfer
  - Any VS Code extension for IBM i development (ie. Code for IBM I and Project Explorer)
  - RDi

Project Structure (Source Code)

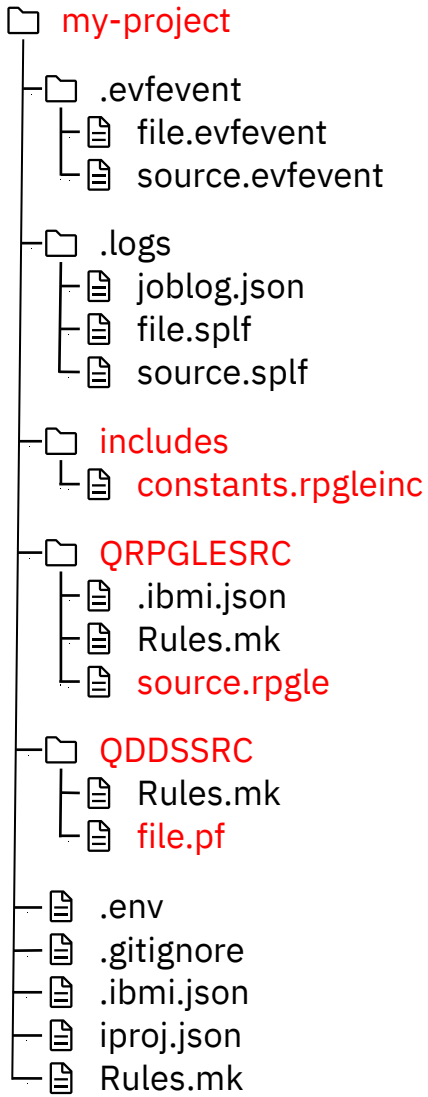


MYLIB

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

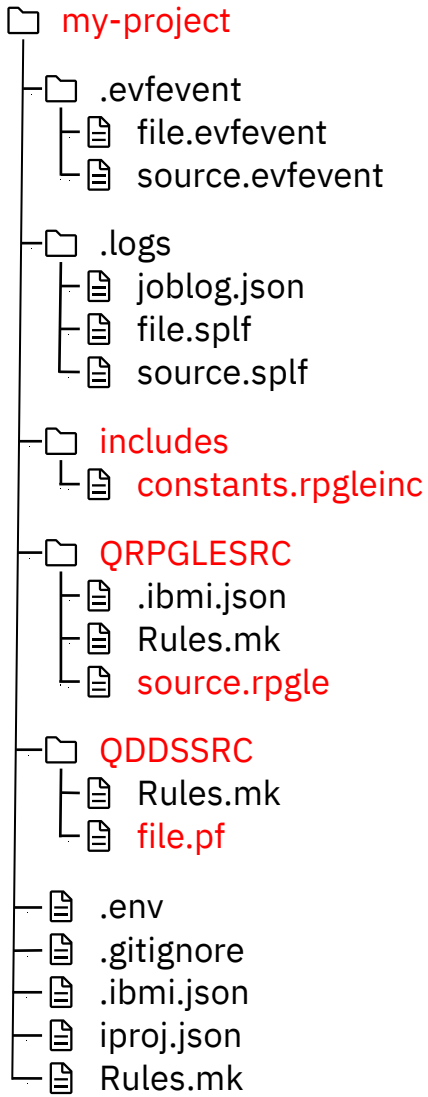
Project Structure (Metadata)

IBM i
continuous innovation
continuous integration



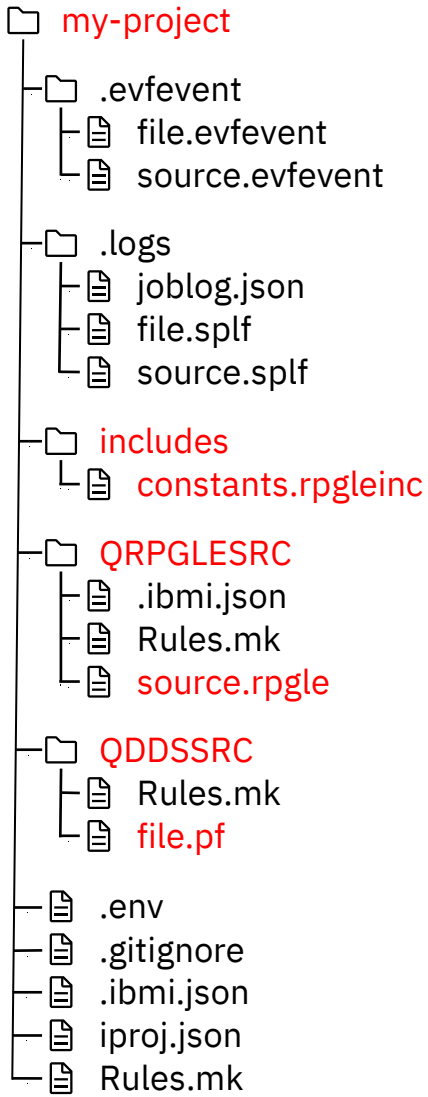
Project Structure (Build/Compile Output)

IBM i
continuous innovation
continuous integration



Project Structure (Rules.mk)

IBM i
continuous innovation
continuous integration



Build and Compile Process

IBM i
continuous innovation
continuous integration

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

- A

IBM i
continuous innovation
continuous integration

Local Development

Different (But Similar) File System

IBM i
continuous innovation
continuous integration

MYLIB
- QRPGLESRC
  - PROGRAMA.RPGLE
  - PROGRAMB.RPGLE
  - PROGRAMC.RPGLE
- QSQLSRC
  - CUSTOMERS.SQL
  - INVENTORY.SQL
- QCLLESRC
  - STARTJOB.CLLE
- QCMDSRC
  - STARTJOB.CMD

/COPY and /INCLUDE

IBM i
continuous innovation
continuous integration

Distributed Development

IBM i
continuous innovation
continuous integration

Version Control with Git

IBM i
continuous innovation
continuous integration

Development with Git

IBM i
continuous innovation
continuous integration

IBM i
continuous innovation
continuous integration

IBM i Project Explorer

Overview

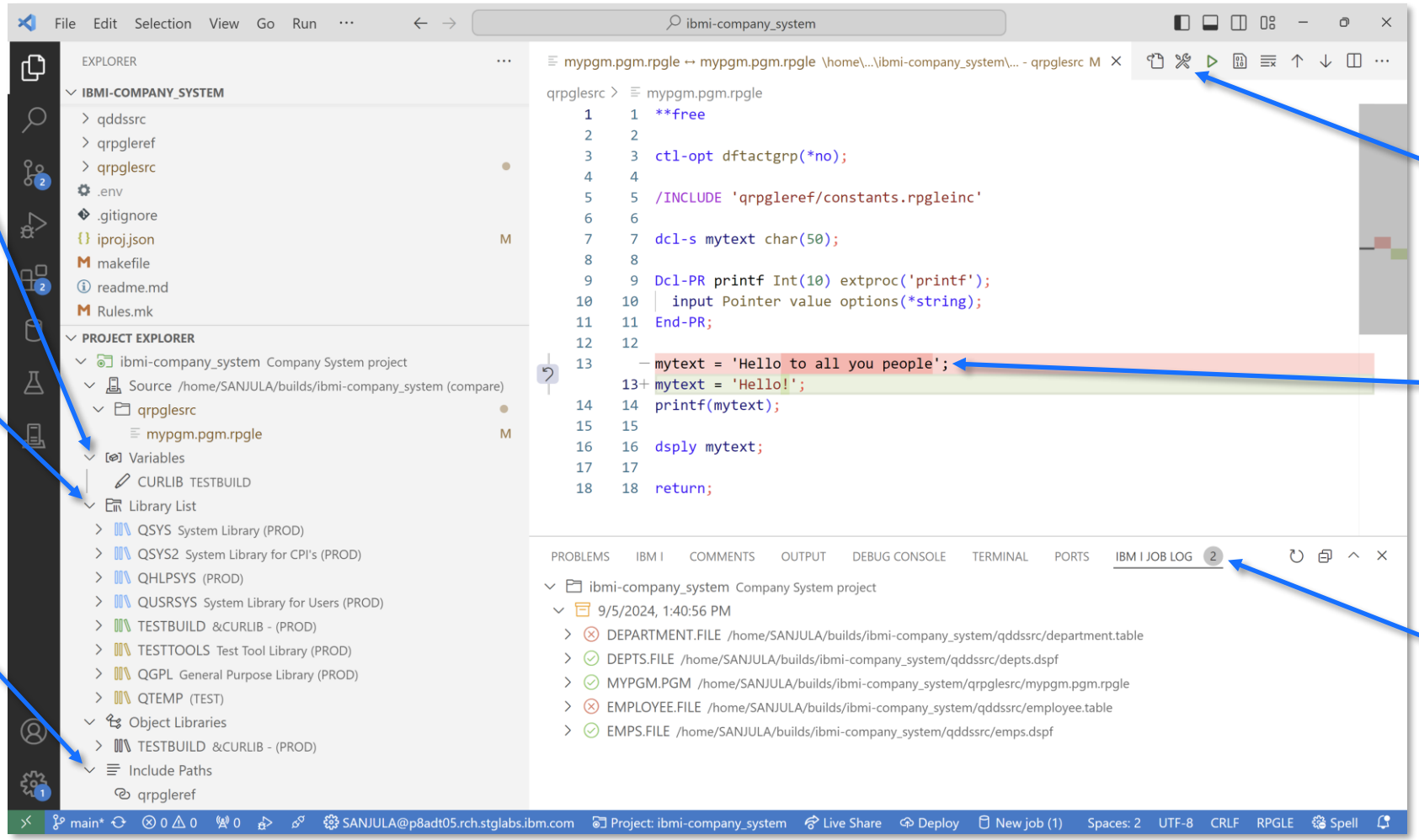
IBM i
continuous innovation
continuous integration

The ultimate tool for local development on IBM i!

Set variables

Manage library list

Modify include paths



Build and Compile

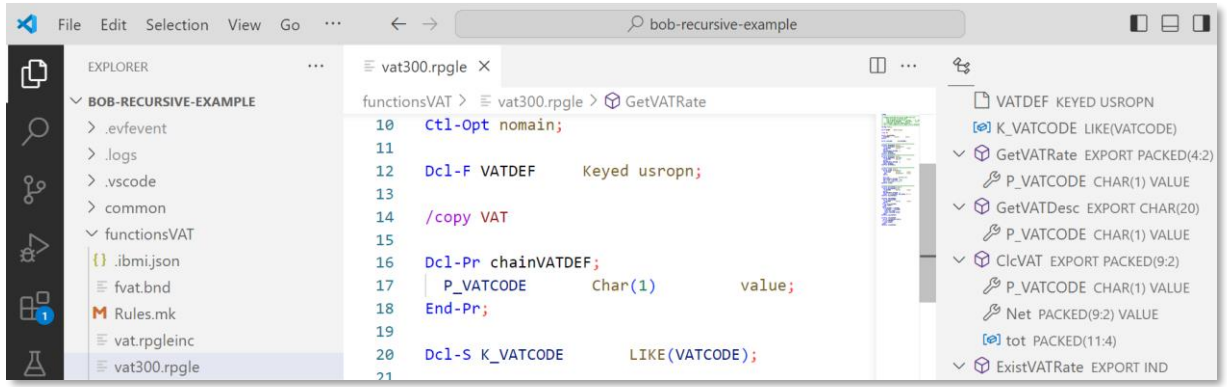
Local source vs. IFS source

View job logs

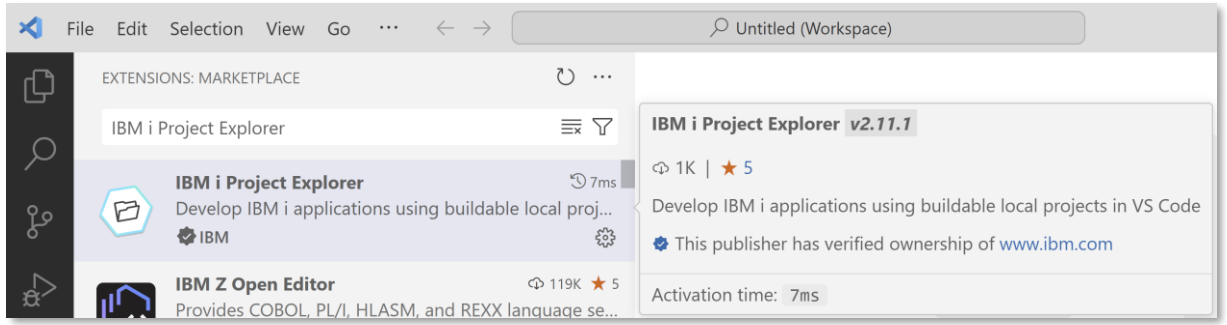
Installation

IBM i
continuous innovation
continuous integration

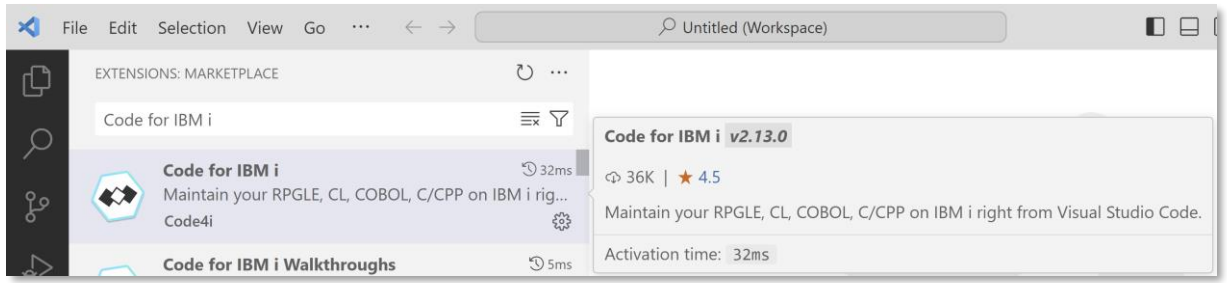
1 *Download*
Visual Studio Code



2 *Download*
IBM i Project Explorer



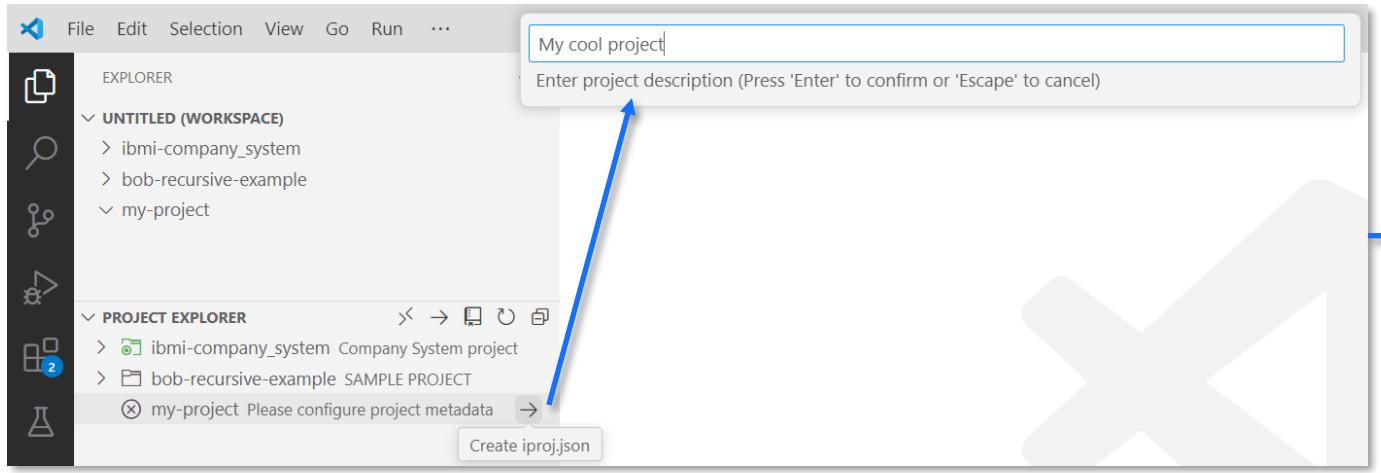
3 *Download*
Code for IBM i

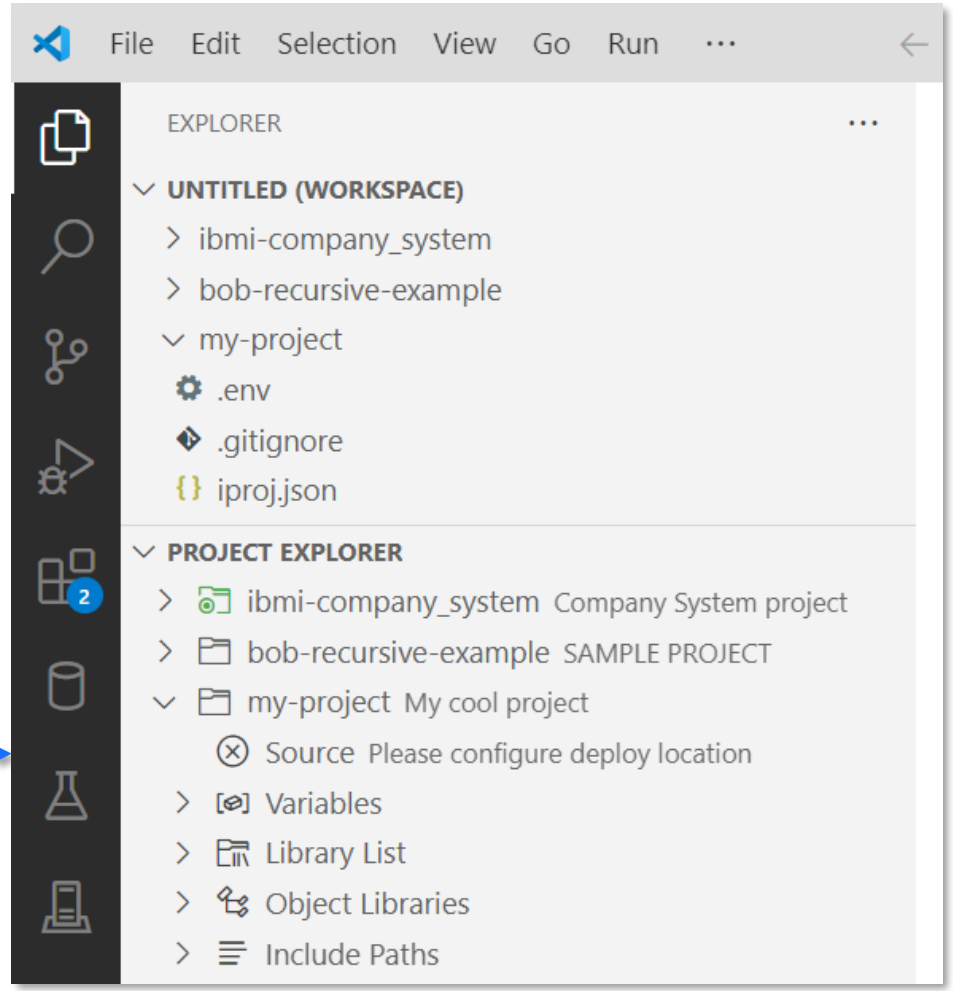


Create a New Project

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

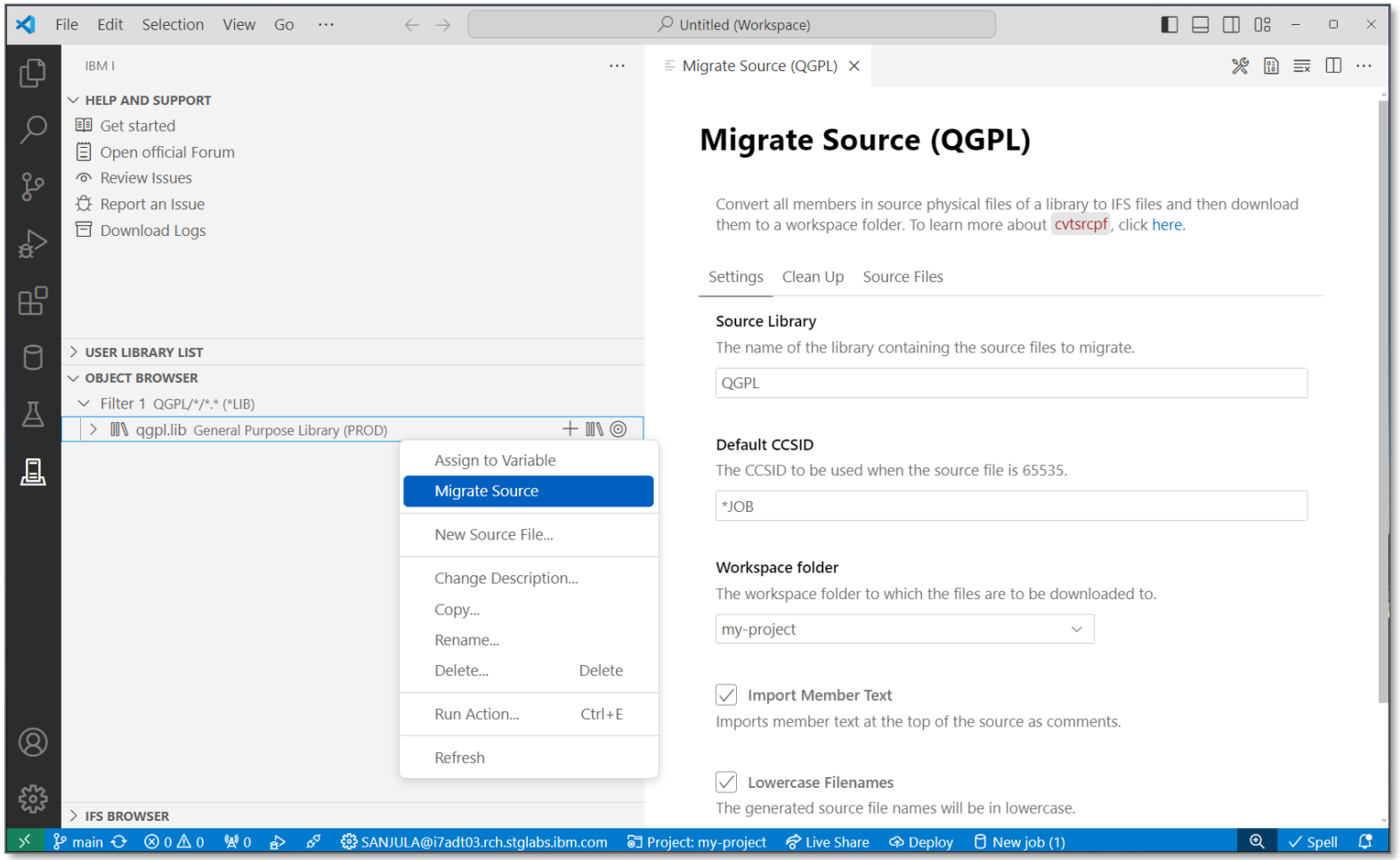
↓

Source Orbit

Rename extensions

↓

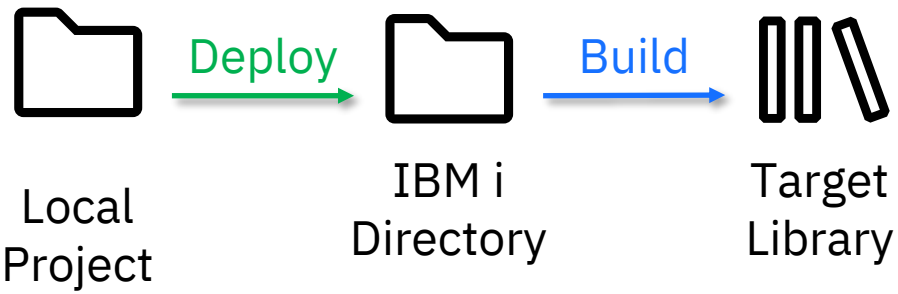
Convert includes/copy directives to Unix style paths

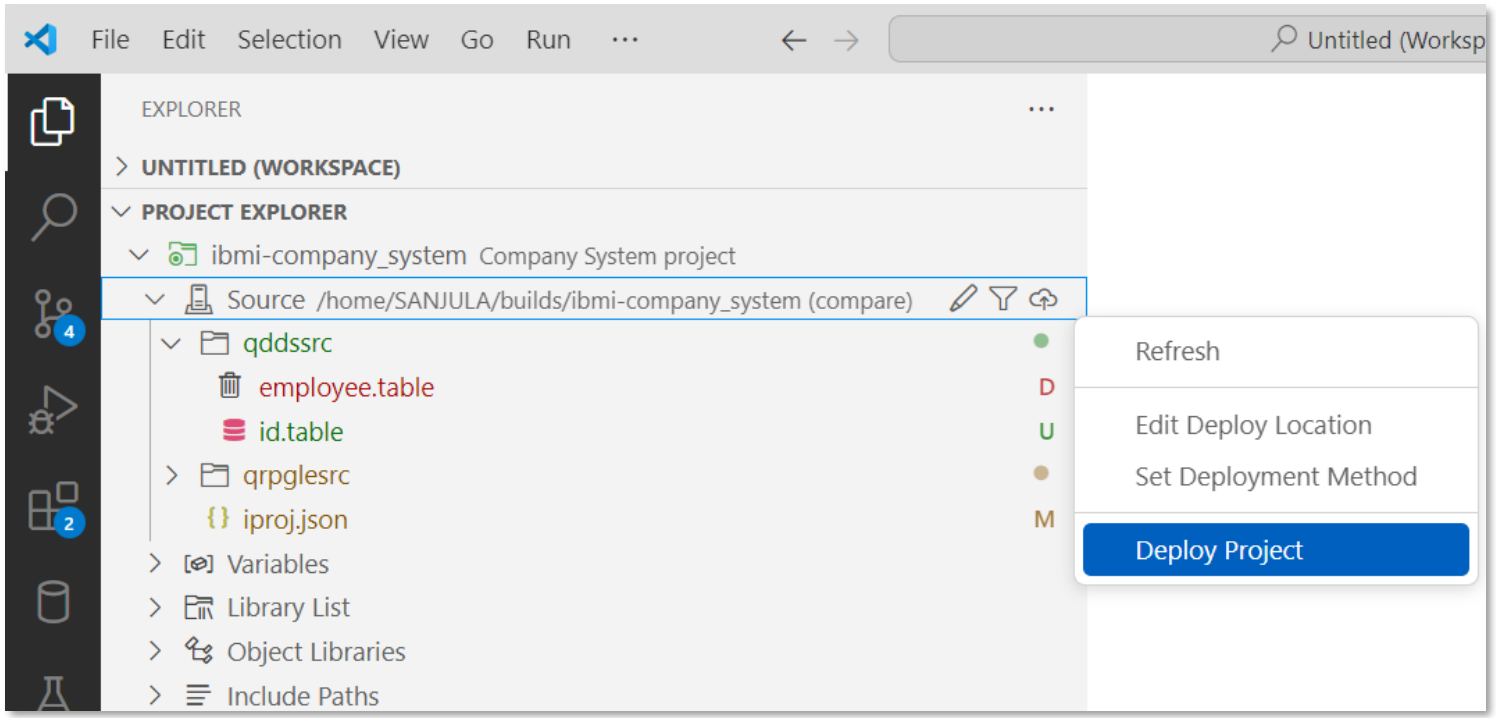


Source and Deployment

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
  - Changes
  - Working Changes
  - Staged Changes
  - All
- Deploy project
  - Moves files to deploy location based on deployment method



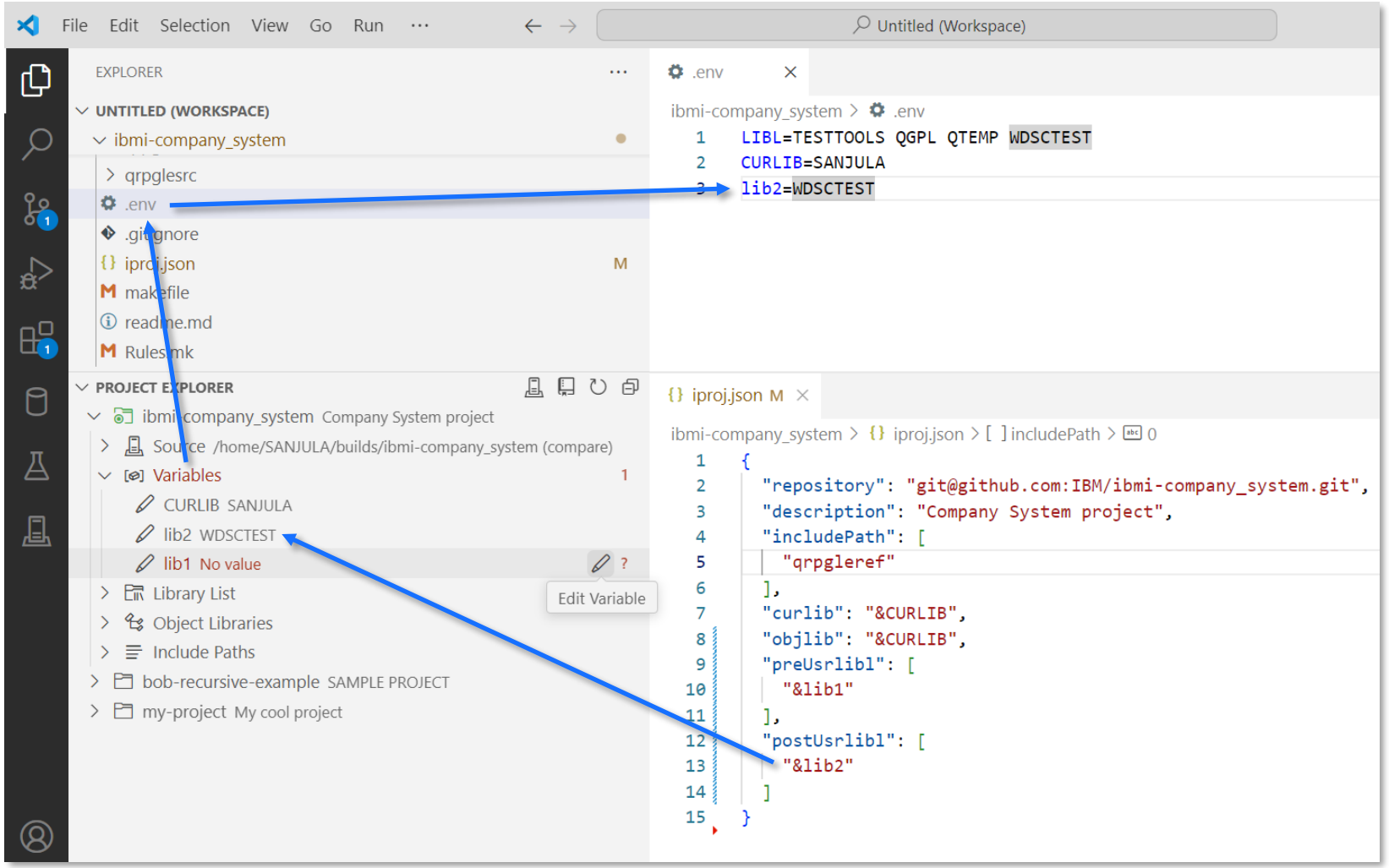


Work with Variables

IBM i
continuous innovation
continuous integration

- Manage variables
- Browse for libraries and assign values to variables
- Configure hardcoded values as variables

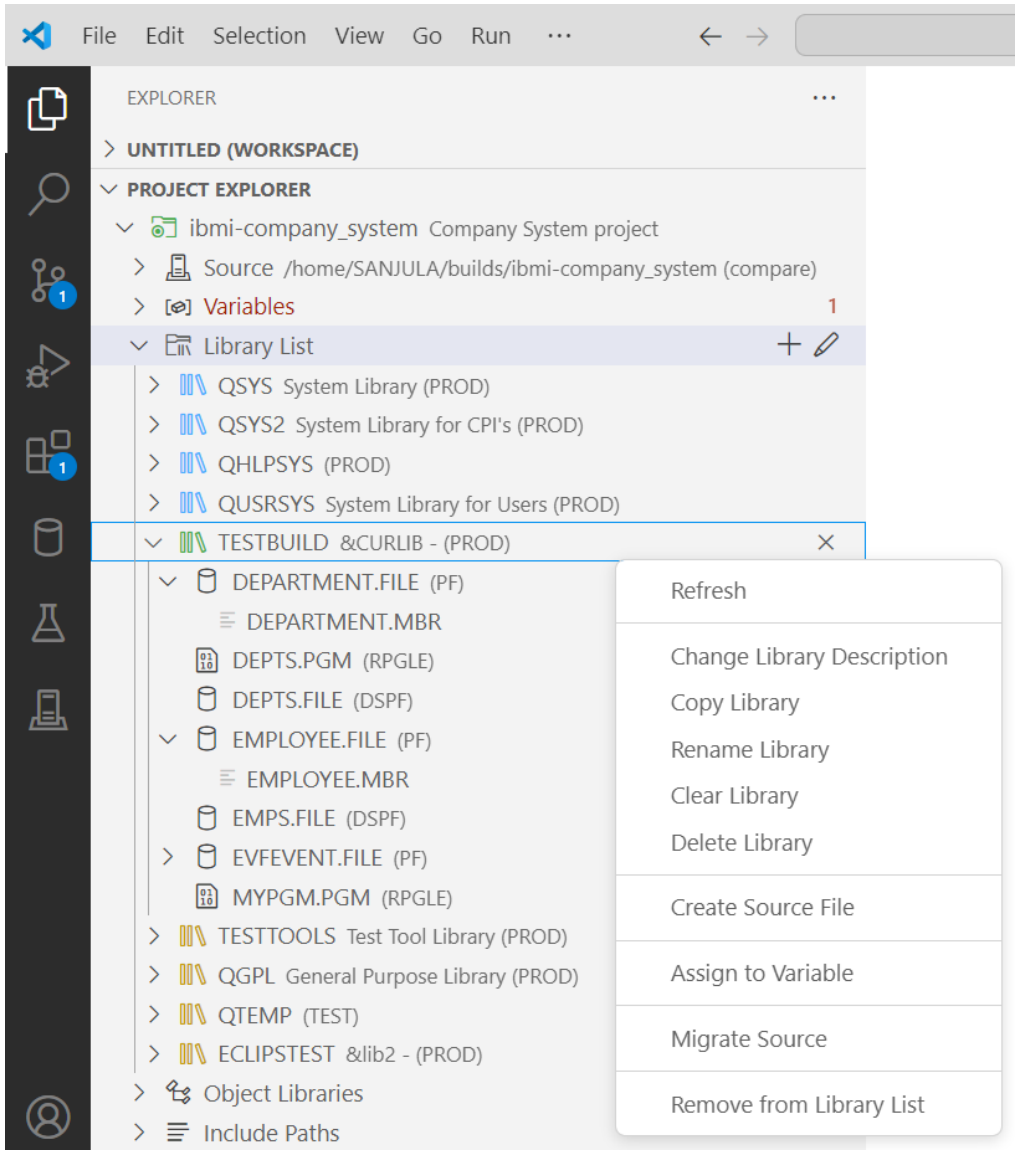
Do not push .env file to Git!



Manage the Library List

IBM i
continuous innovation
continuous integration

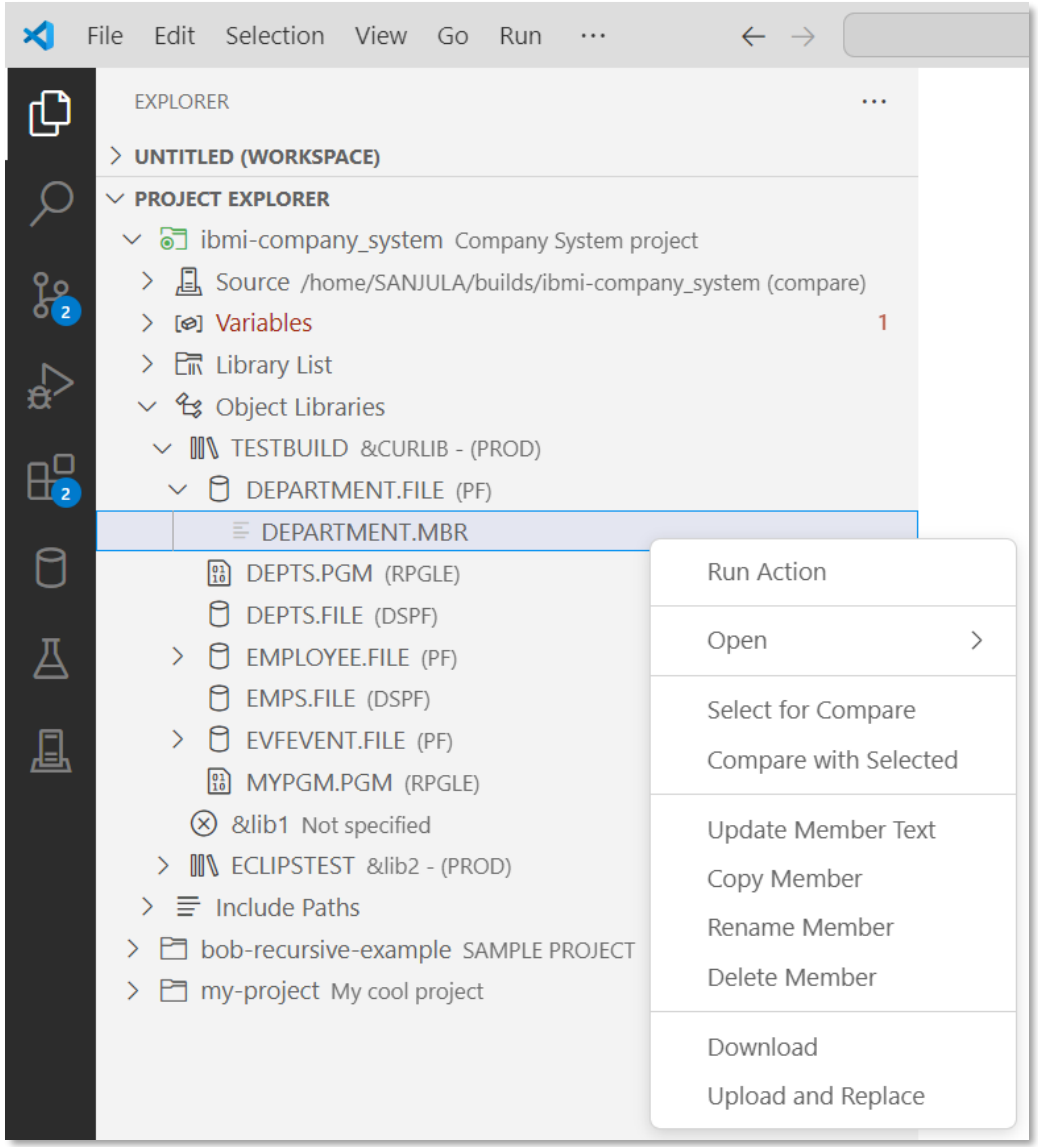
- Add to library list (preUsrlibl and postUsrlibl in iproj.json)
- Set current library (curlib in iproj.json)
- Reorder library list
- Browse objects and members
- Manage libraries, objects, and members



Browse Object Libraries

IBM i
continuous innovation
continuous integration

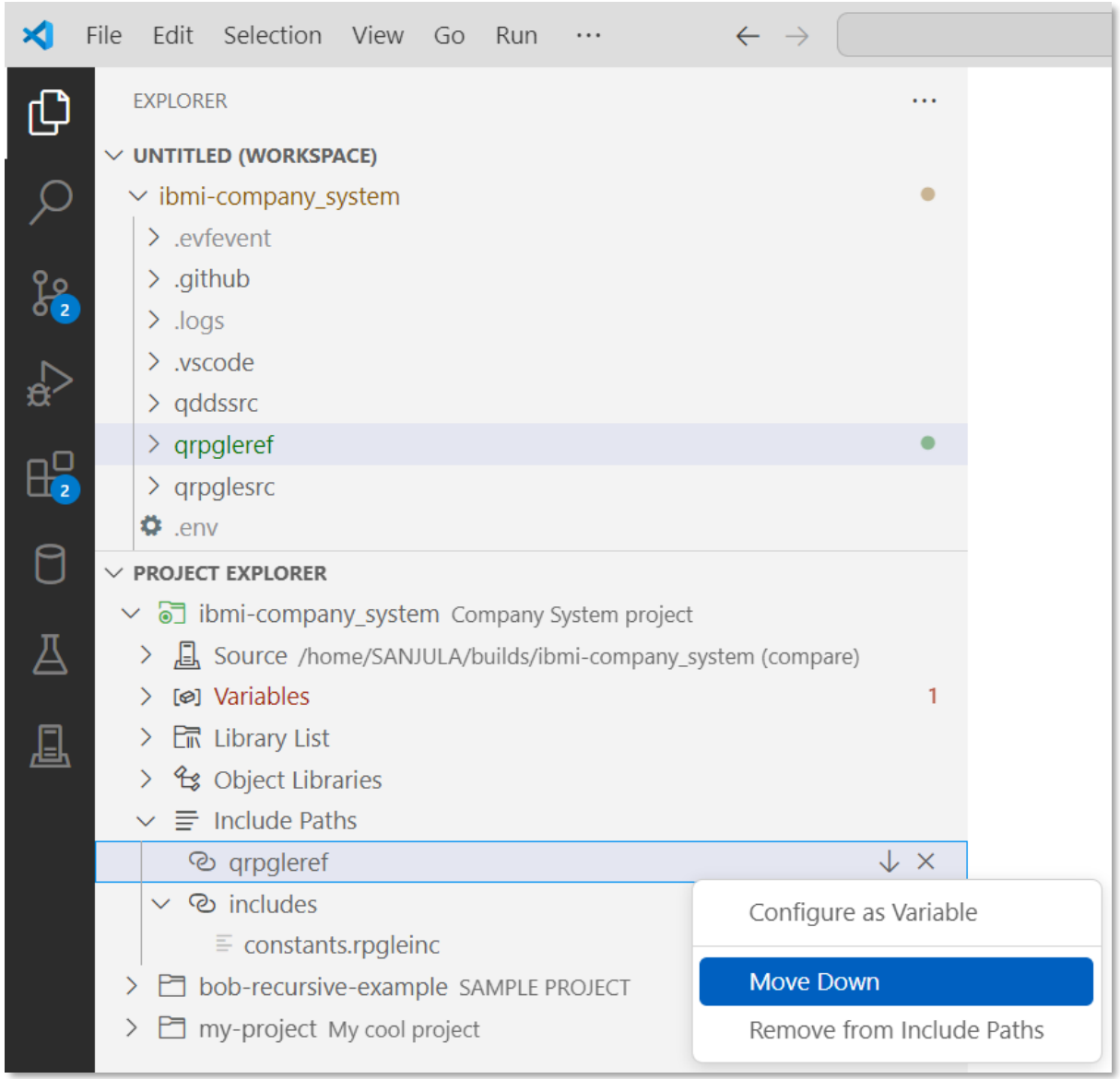
- Another place to manage libraries in iproj.json (curlib, objlib, preUsrlibl, postUsrLibl)
- Manage libraries, objects, and members



Manage Include Paths

IBM i
continuous innovation
continuous integration

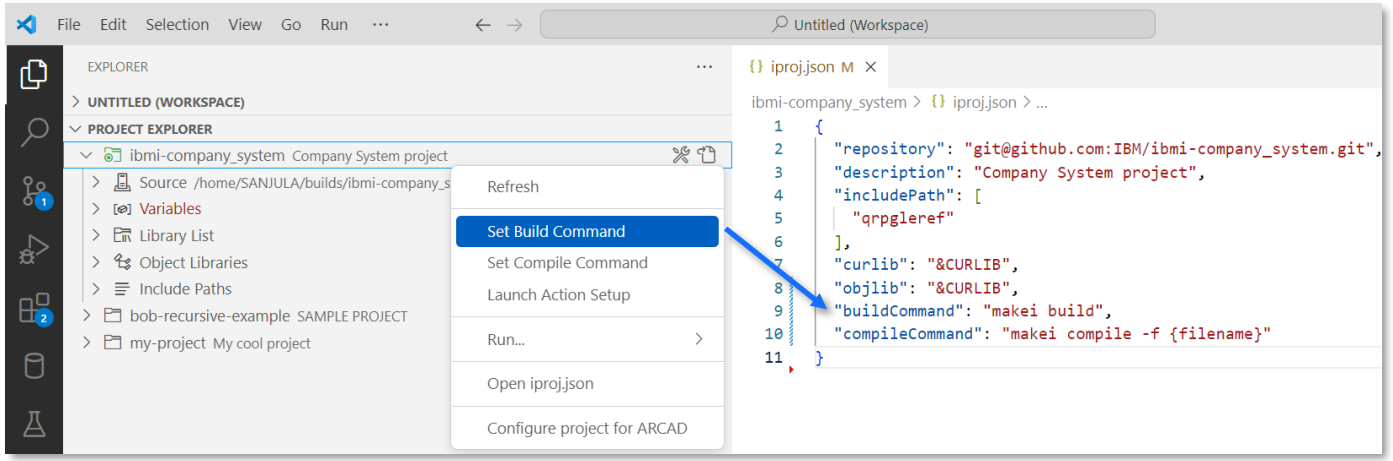
- Add, remove, and reorder include paths
- Visualize if includes resolve locally or to remote IFS



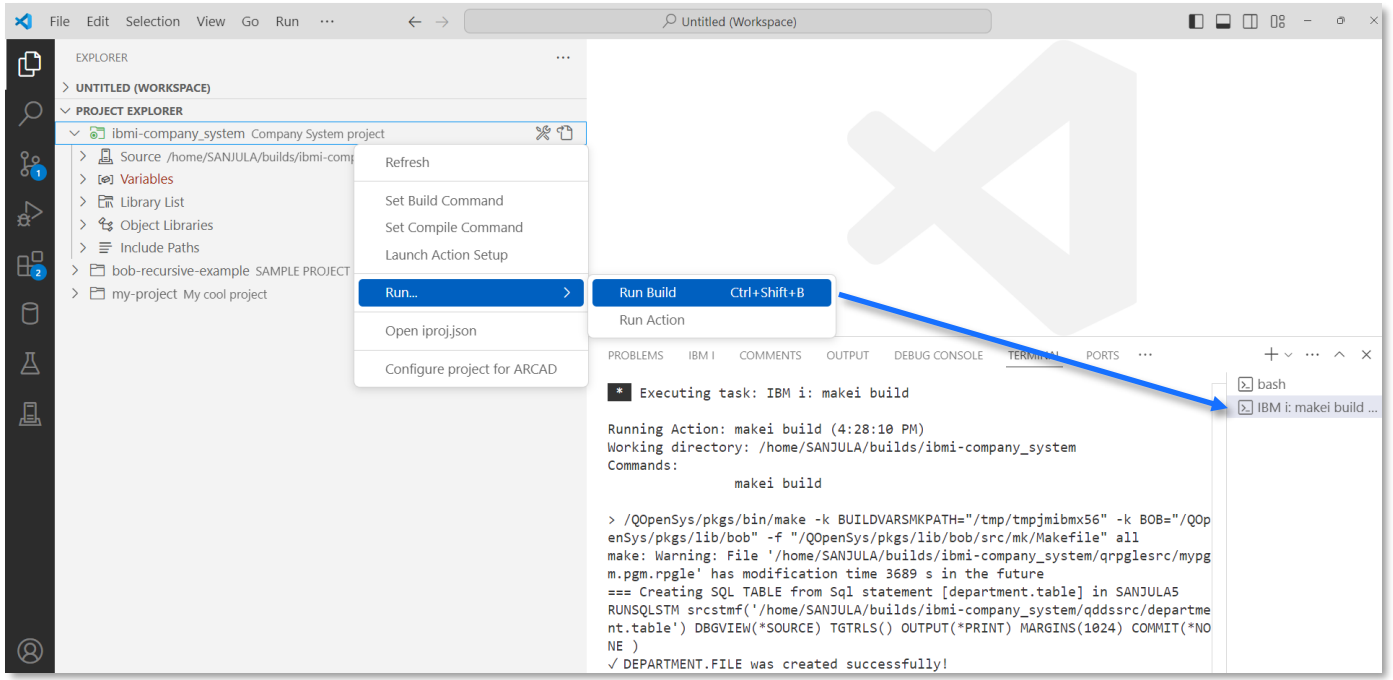
Building and Compiling

IBM i
continuous innovation
continuous integration

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



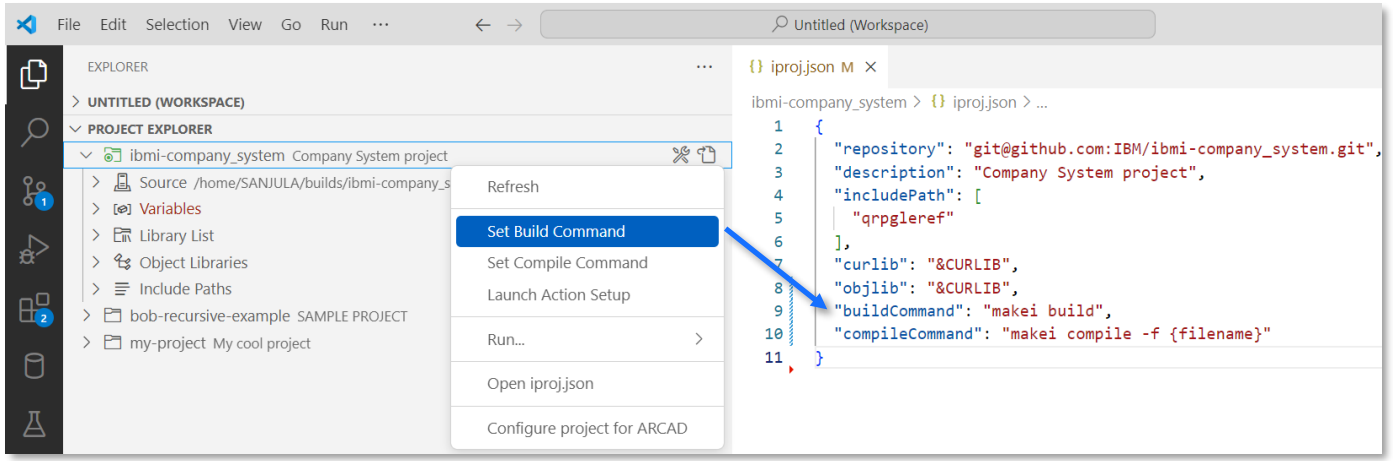
- Building
  - Set build command
  - Run Build
- Compiling
  - Set compile command
  - Run compile
    - On active editor
    - On file or directory in File Explorer
    - On file or directory in Source



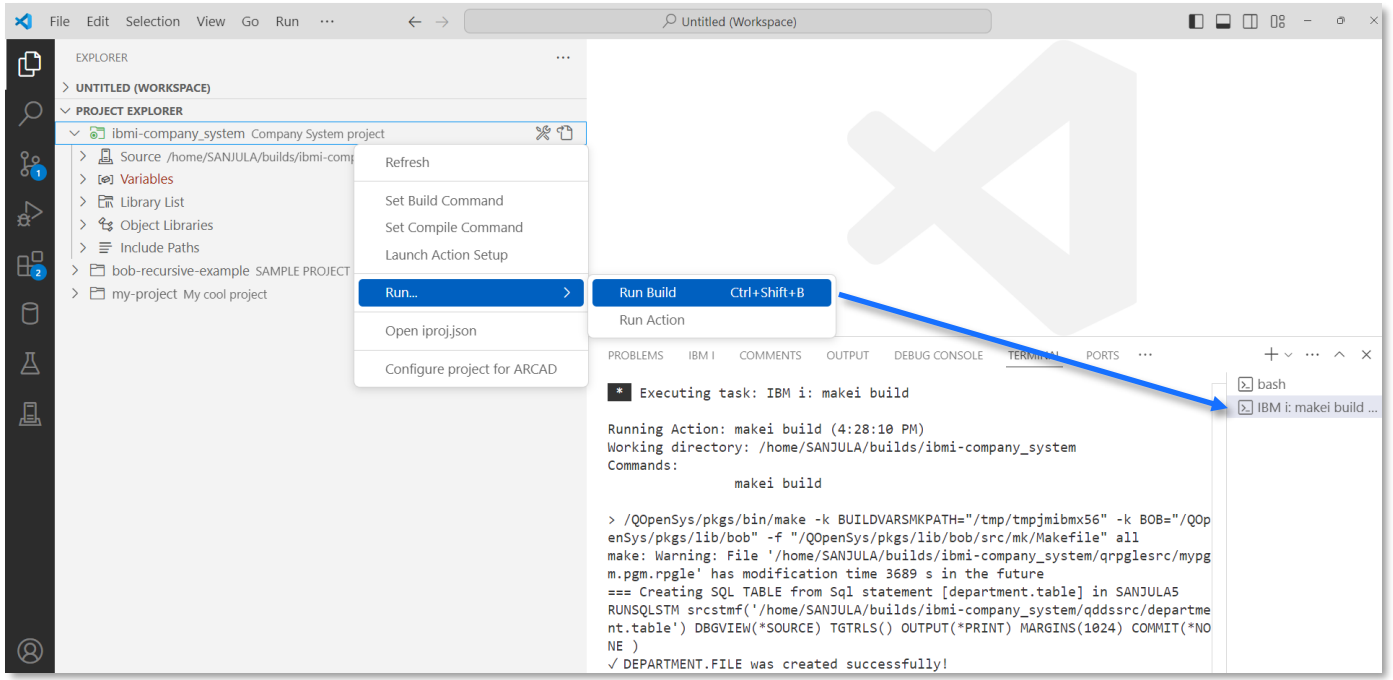
Running Actions

IBM i
continuous innovation
continuous integration

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



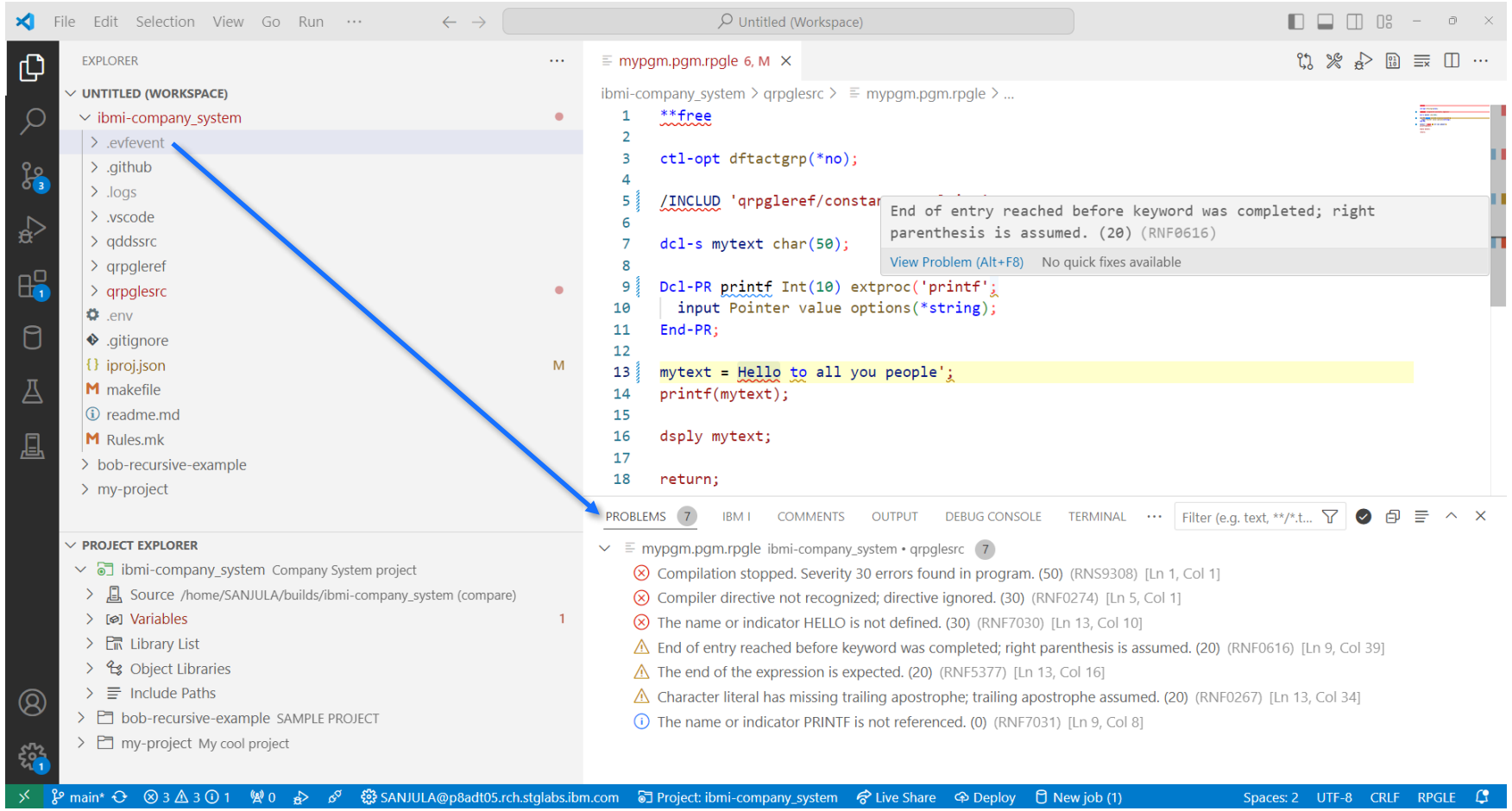
- Building
  - Set build command
  - Run Build
- Compiling
  - Set compile command
  - Run compile
    - On active editor
    - On file or directory in File Explorer
    - On file or directory in Source



View Diagnostics

IBM i
continuous innovation
continuous integration

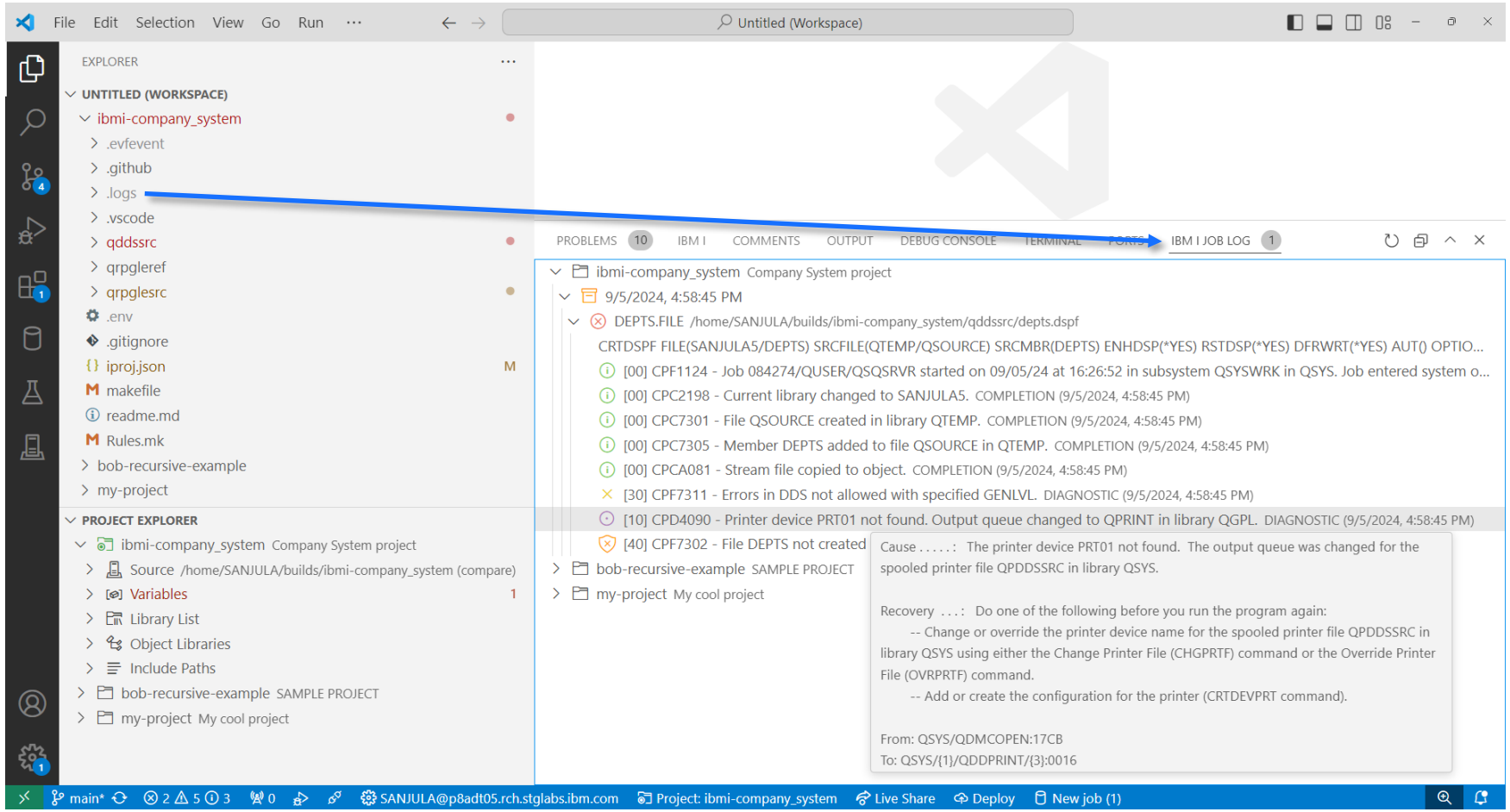
- Evfevent file diagnostics are dumped in .evfevent directory after a build or compile
- Code for IBM i parses and renders them in the Problems view
- Diagnostics are also rendered inline in the source file



View Job Logs

IBM i
continuous innovation
continuous integration

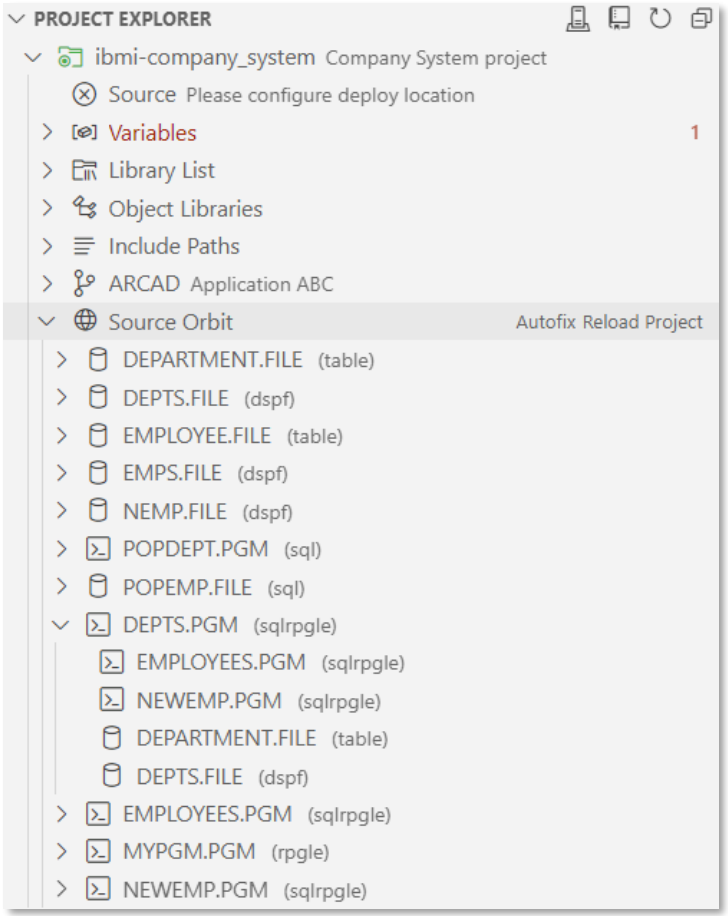
- Job log and spool files are dumped in .logs directory after a build or compile
- Job log view is used to visualize and mangae these logs
- Track up to 10 of the previous logs in memory
- Organized by the ILE objects being built
- Filter by failed objects or severity



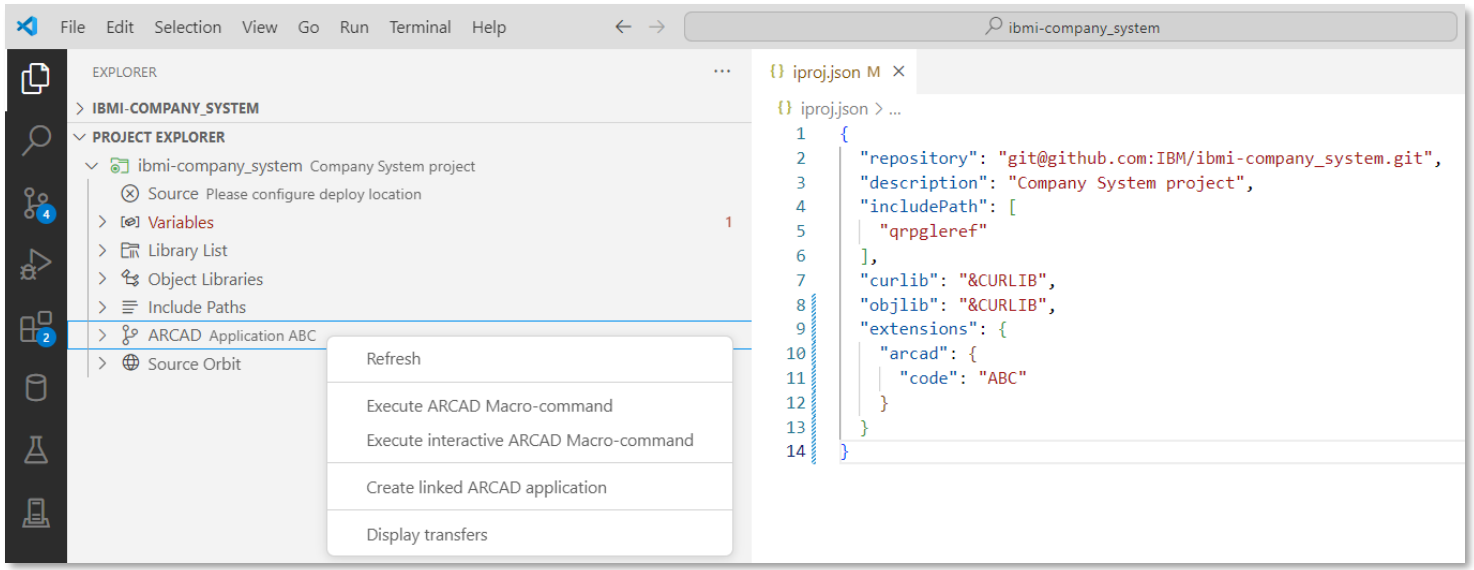
Integration

IBM i
continuous innovation
continuous integration

Source Orbit



ARCAD-Elias



What can you integrate?

Demo