Modern Buildable Projects with IBM i Project Explorer and Bob

Edmund Reinhardt, IBM
Product Architect – IBM i App Dev & AI Solutions for IBM i
edmund.reinhardt@ca.ibm.com

Sanjula Ganepola, IBM
Software Developer – IBM i App Dev & AI Toolchain
Sanjula.Ganepola@ibm.com

Agenda

IBM i
continuous innovation
continuous integration

- Challenges with Building on IBM i
- How does local development overcome this?
- What are IBM i Projects?
- What is Bob and how to use it?
- Ins and Outs of IBM i Project Explorer

IBM i
continuous innovation
continuous integration

Challenges with Building on IBM i

Building on IBM i is limiting...

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

Let’s look ahead to the future

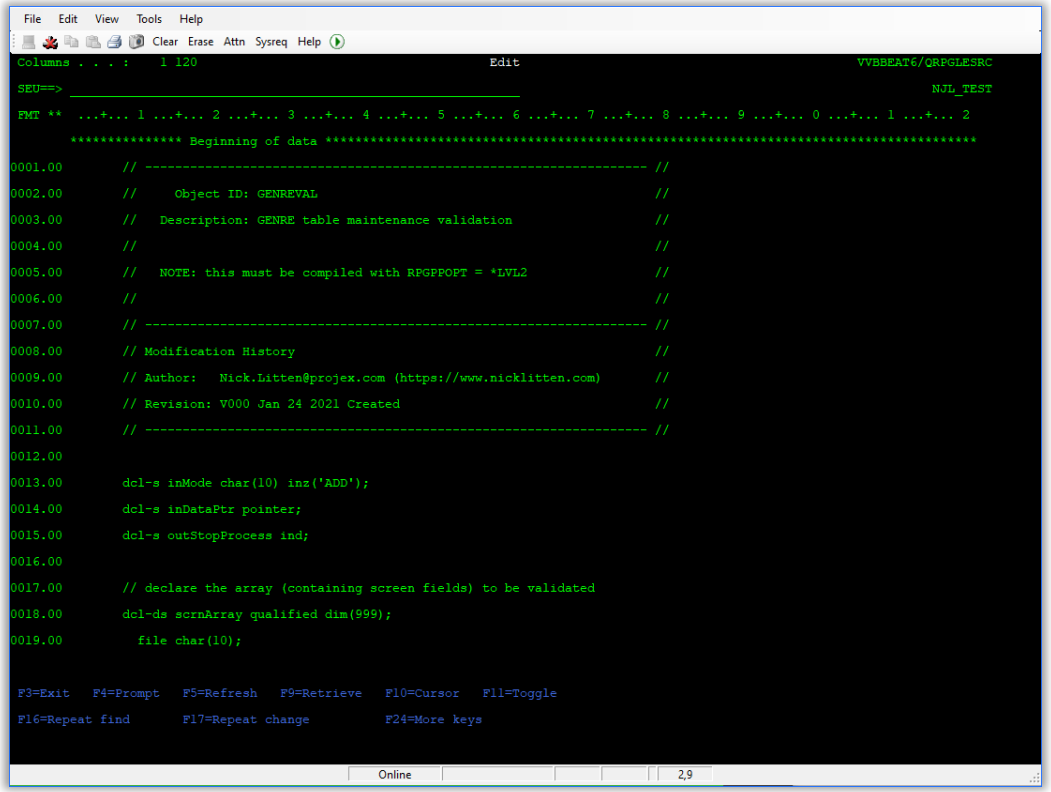
IBM i
continuous innovation
continuous integration

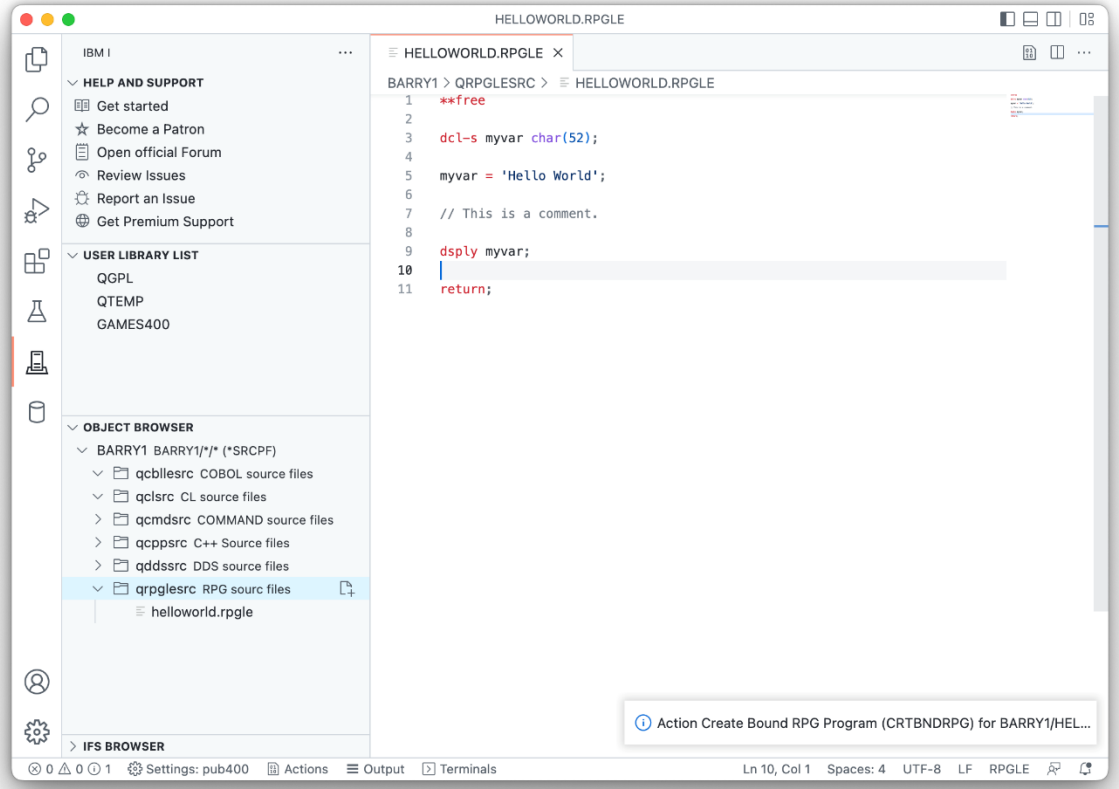
New Developers Incoming!

Need for Modernization is Urgent

New development tools/ecosystems (ie. Code for IBM i/Merlin)

Modern development practices (ie. Git)





How does local development overcome this?

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

IBM i
continuous innovation
continuous integration

MYPROJECT
- QRPGLESRC
  - CUSTOMER.RPGLE
  - INVENTORY.RPGLE
  - STARTJOB.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

/my-project
- /.git
- qrpglesrc
  - customer.rpgle
  - inventory.rpgle
  - startjob.rpgle
- qsqlsrc
  - customer.table
  - inventory.table
- qcllesrc
  - start.clle
- qcmdsrc
  - startjob.cmd

IFS/Local File System

Let’s go one step further!

IBM i
continuous innovation
continuous integration

/my-project
- /.git
- qrpglesrc
  - customer.rpgle
  - inventory.rpgle
  - startjob.rpgle
- qsqlsrc
  - customer.table
  - inventory.table
- qcllesrc
  - start.clle
- qcmdsrc
  - startjob.cmd

IFS/Local File System

Flexible directory structure

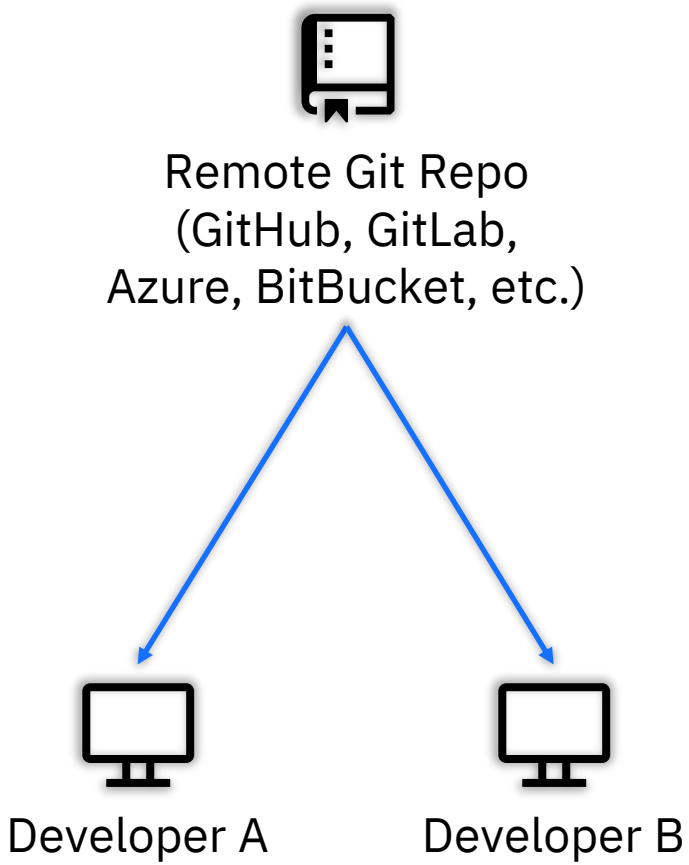
/my-project
- /.git
- customer-management
  - customer.rpgle
  - customer.table
- inventory-service
  - inventory.rpgle
  - inventory.table
- start-command
  - startjob.pgm.rpgle
  - start.clle
  - startjob.cmd

Logical Organization

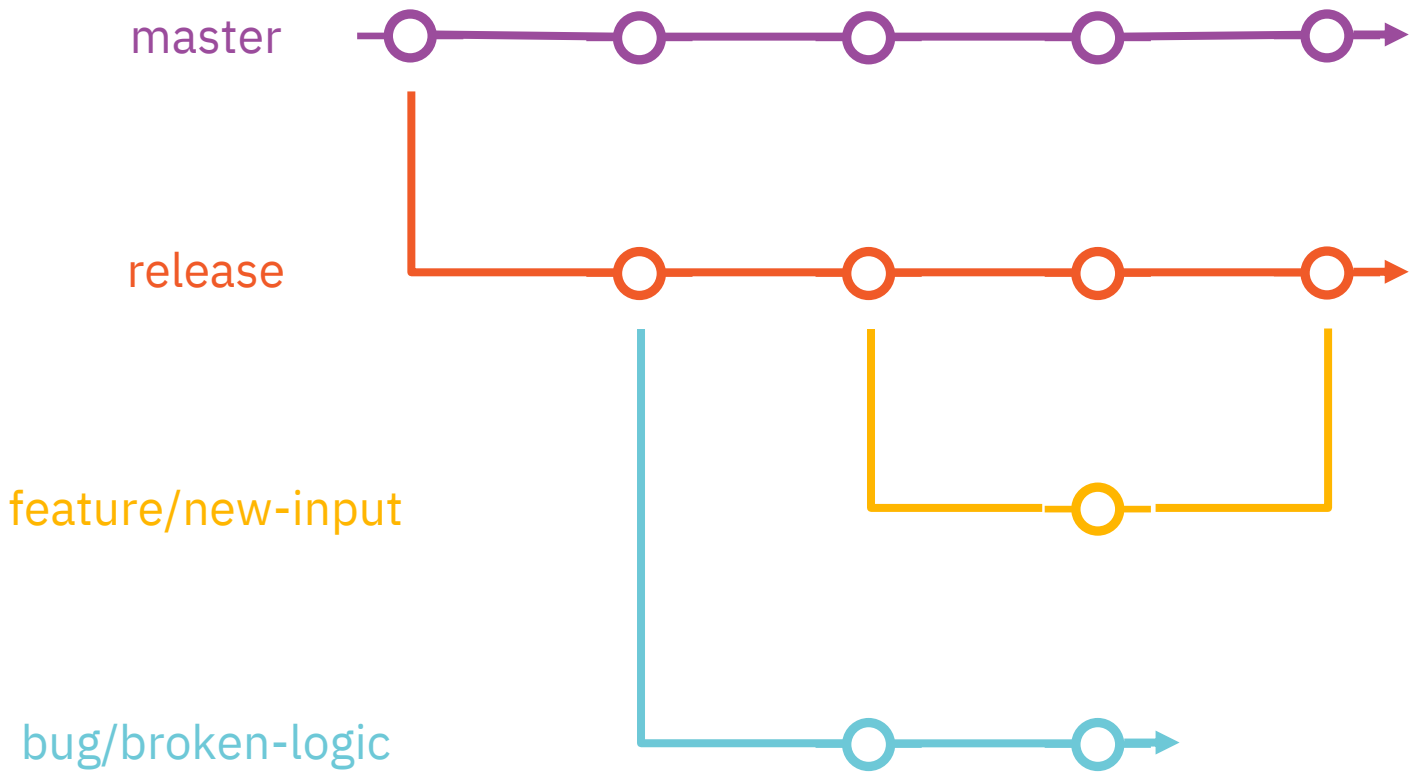
Unlocking source control with Git

IBM i
continuous innovation
continuous integration

Distributed Development



Version Control and Git Workflow



Did we solve our problems?

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

IBM i
continuous innovation
continuous integration

What are IBM i Projects?

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"
}
```

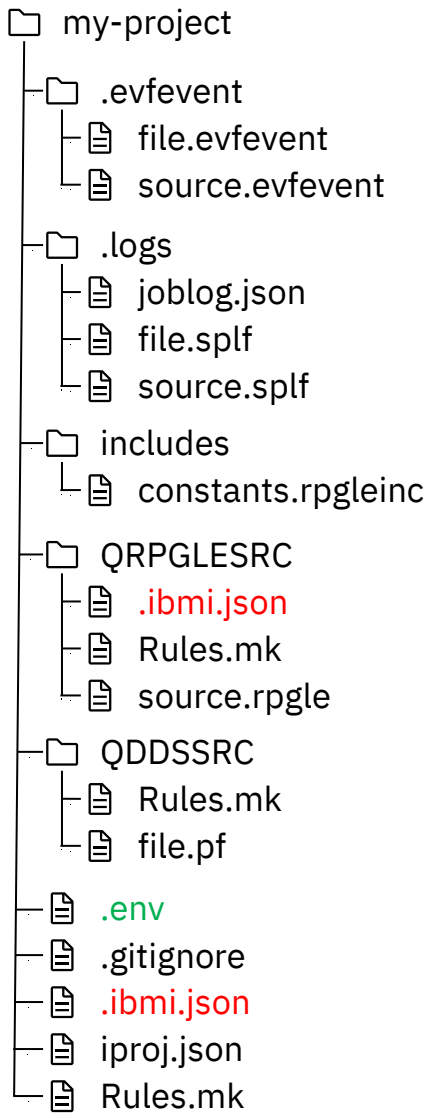
Standardized metadata format with variables (&...)

Set build/compile command

iproj.json in project root

Flexible subdirectories and build customization

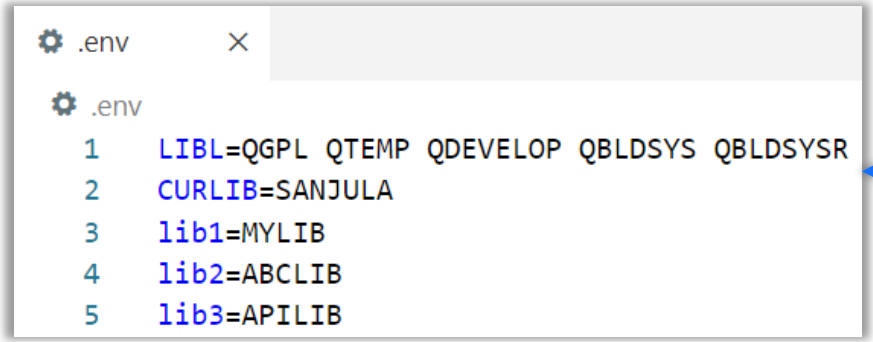
IBM i
continuous innovation
continuous integration



EBCDIC encoding for compiler

Target object library for directory

.ibmi.json in project root or subdirectories



Custom variable values so that each developer can customize build

.env in project root

IBM i
continuous innovation
continuous integration

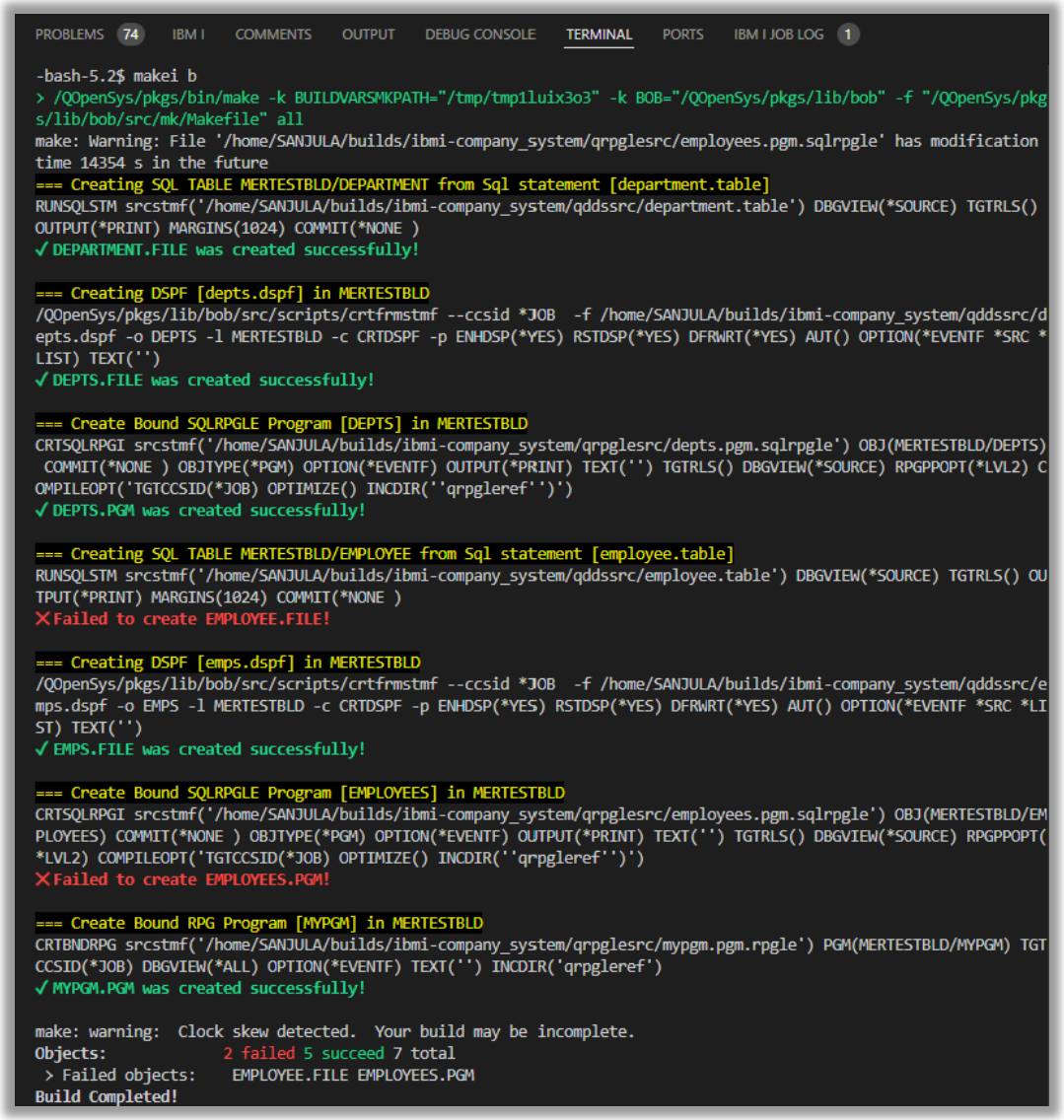
What is Bob and how to use it?

What is Bob?

IBM i
continuous innovation
continuous integration

Free and open-source build system to build QSYS objects on IBM i

- 🚀 Speed: Compile objects that need recompiling (new or changed source code)
- 🛠️ Reliability: If an item changes, then it and everything it depending on it will be rebuilt
- 💎 Industry standard: Object dependencies are specified using standard makefile syntax
- 🔧 Flexibility: Override compile parameters, write custom recipes (If you can code it, you can build it!)
- 🌟 Ease of use: Build with a single command or a single button in an IDE (IBM i Project Explorer in VS Code)



Defining targets with Rules.mk

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: private TEXT = Functions VAT
3
4   VAT300.MODULE: vat300.rpgle QPROTOSRC/vat.rpgleinc VATDEF.FILE
5   VAT300.MODULE: private TEXT := bound into FVAT.SRVPGM
6   VAT300.MODULE: private DBGVIEW ::= *SOURCE
7
8   VATDEF.FILE: vatdef.pf SAMREF.FILE
```

Makefile with list of objects to be built and from which source files

Rules.mk in subdirectories

Variables in Rules.mk

IBM i
continuous innovation
continuous integration

Declare variables to override compile settings for multiple targets

Rules.mk U

SAMPLE > Rules.mk

```
1   PRO200.MODULE: private TGTRLS := *PRV
2   PRO200.MODULE: PRO200.RPGLE
3
4   VAT.MODULE: private TGTRLS := *PRV
5   VAT.MODULE: VAT.RPGLE
```


Rules.mk U

SAMPLE > Rules.mk

```
1   PROJECT_TGTRLS := *PRV
2
3   PRO200.MODULE: private TGTRLS := $(PROJECT_TGTRLS)
4   PRO200.MODULE: PRO200.RPGLE
5
6   VAT.MODULE: private TGTRLS := $(PROJECT_TGTRLS)
7   VAT.MODULE: VAT.RPGLE
```

Wildcarding in Rules.mk

IBM i
continuous innovation
continuous integration

Use wildcarding when you have files that create objects of the same type

Rules.mk

```
1   FILE_TGTRLS = V7R3M0
2   MOD_TGTRLS = V7R3M0
3
4   TEST1.FILE: private TGTRLS := $(FILE_TGTRLS)
5   TEST1.FILE: TEST1.TABLE DEP1.FILE DEP2.FILE
6
7   TEST2.FILE: private TGTRLS := $(FILE_TGTRLS)
8   TEST2.FILE: TEST2.TABLE DEP1.FILE DEP2.FILE
9
10  TEST1.MODULE: private TGTRLS := $(MOD_TGTRLS)
11  TEST1.MODULE: TEST1.RPGLE
```


Rules.mk

```
1   FILE_TGTRLS = V7R3M0
2   MOD_TGTRLS = V7R3M0
3
4   %.FILE: private TGTRLS := $(FILE_TGTRLS)
5   %.MODULE: private TGTRLS := $(MOD_TGTRLS)
6
7   %.FILE: %.TABLE DEP1.FILE DEP2.FILE
8   %.MODULE: %.RPGLE
```

Overriding Wildcards in Rules.mk

IBM i
continuous innovation
continuous integration

Explicitly defined compile options take precedence over wildcarding

Rules.mk

```
1   FILE_TGTRLS = V7R3M0
2   MOD_TGTRLS = V7R3M0
3
4   %.FILE: private TGTRLS := $(FILE_TGTRLS)
5   %.MODULE: private TGTRLS := $(MOD_TGTRLS)
6
7   %.FILE: %.TABLE DEP1 DEP2
8   %.MODULE: %.RPGLE
9
10  EMP.MODULE: private TGTRLS := V7R4M0
11  EMP.MODULE: EMP.RPGLE A.TABLE
```

Supported Object Types

IBM i
continuous innovation
continuous integration

These IBM i source types can be compiled directly from the IFS ...

Object Type	File Extension
*CMD	.CMD
*MODULE	.RPGLE, .CLLE, .C, .SQLC, .CPP, .SQLCPP, .SQLRPGLE, .CBLLE, .SQLCBLLE
*PGM	.PGM.RPGLE, .PGM.SQLRPGLE, .PGM.C, .PGM.CBLLE, .PGM.SQLCBLLE
*SRVPGM	.BND

These older IBM i source types can be compiled as well (CRTFRMSTMF copies the source to QTEMP/QSOURCE before compiling)

Object Type	File Extension
*FILE	.DSPF, .LF, .PF, .PRTF
*MENU	.MENUSRC
*MODULE	.CLLE
*PGM	.RPG, .PGM.CLLE
*PNLGRP	.PNLGRPSRC
*WSCST	.WSCSTSRC
*QMQRY	.SQL

CL pseudo-source

IBM i
continuous innovation
continuous integration

The CL command to create these object types are stored in a file with the given extension ...

Object Type	File Extension	CL Command
*MSGF	.MSGF	CRTMSGF + ADDMSGD
*BNDDIR	.BNDDIR	CRTBNDDIR
*PGM	.ILEPGM	CRTPGM
*SRVPGM	.ILESRVPGM	CRTSRVPGM
*DTAARA	.DTAARA	CRTDTAARA
*DTAQ	.DTAQ	CRTDTAQ
*TRG	.SYSTRG	ADDPFTRG

SQL pseudo-source

IBM i
continuous innovation
continuous integration

The SQL command to create these object types are stored in a file with the given extension and are created via RUNSQLSTM ...

SQL Type	QSYS Object	File Extension	SQL COMMAND
TABLE	*FILE	.TABLE	CREATE OR REPLACE TABLE
VIEW	*FILE	.VIEW	CREATE OR REPLACE VIEW
INDEX	*FILE	.INDEX	CREATE INDEX
PROCEDURE	*PGM	.SQLPRC	CREATE OR REPLACE PROCEDURE
FUNCTION	*SRVPGM	.SQLUDF	CREATE OR REPLACE FUNCTION
FUNCTION	*SRVPGM	.SQLUDT	CREATE DISTINCT TYPE
TRIGGER	*PGM	.SQLTRG	CREATE OR REPLACE TRIGGER
ALIAS	*FILE	.SQLALIAS	CREATE OR REPLACE ALIAS
SEQUENCE	*DTAARA	.SQLSEQ	CREATE OR REPLACE SEQUENCE

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

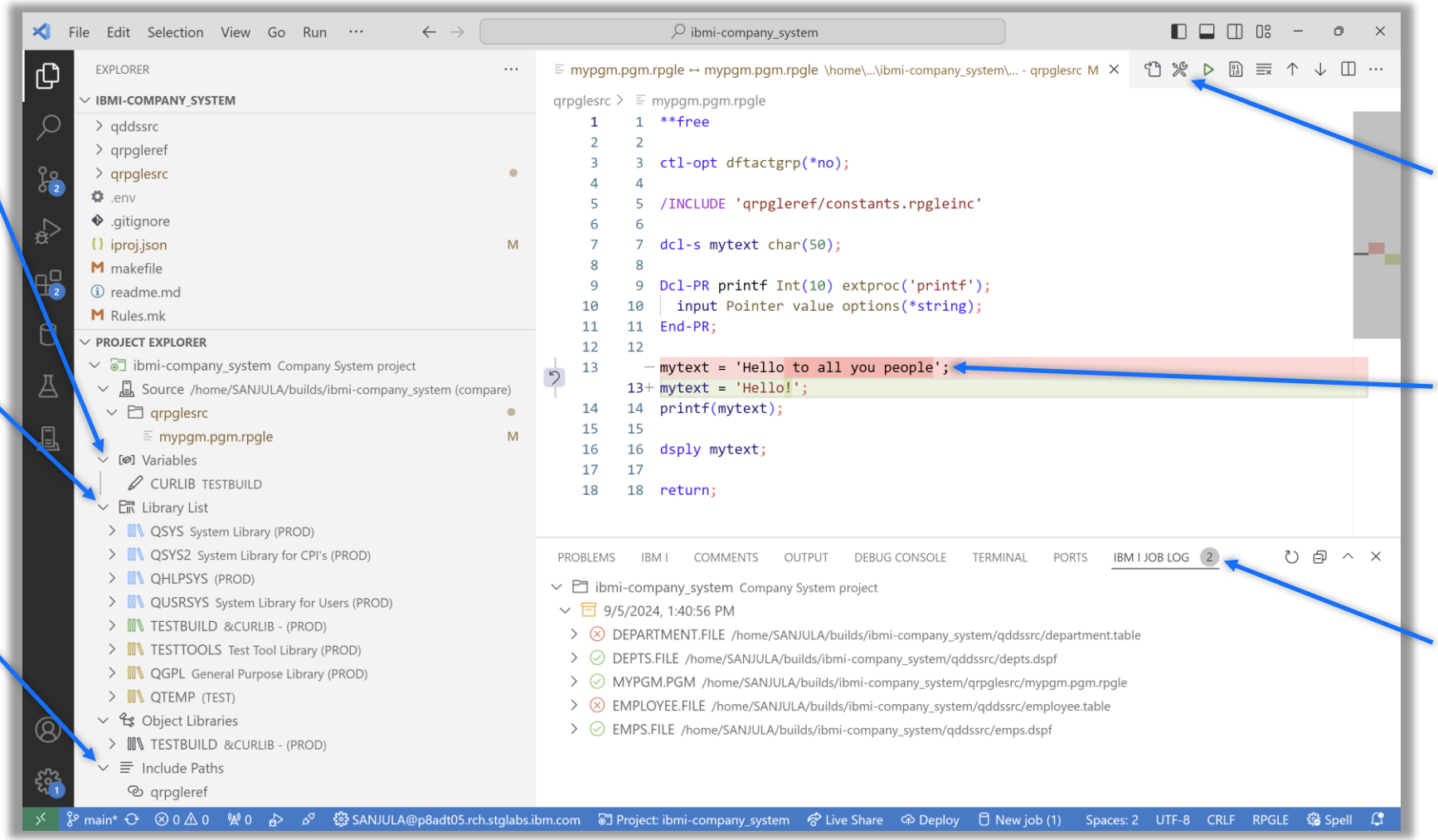
IBM i
continuous innovation
continuous integration

The ultimate tool for local development on IBM i!

Set variables

Manage library list

Modify include paths



Build and Compile

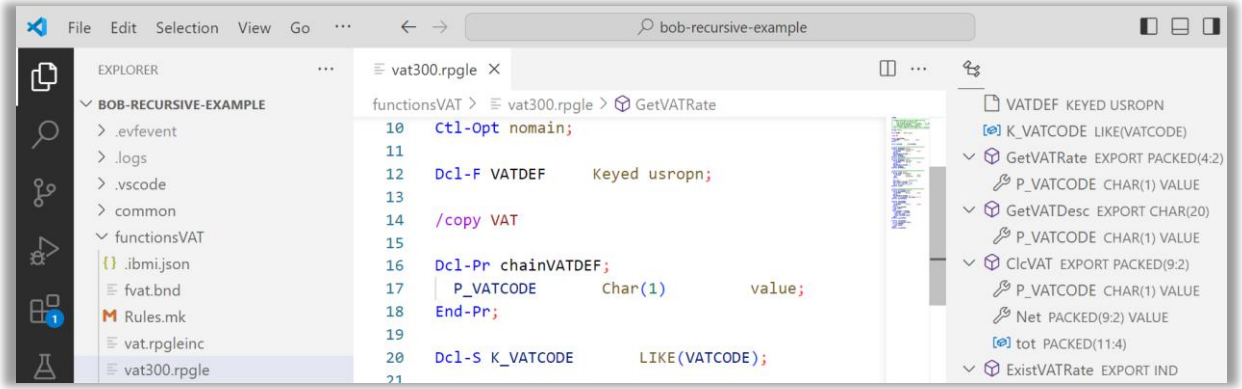
Local source vs. IFS source

View job logs

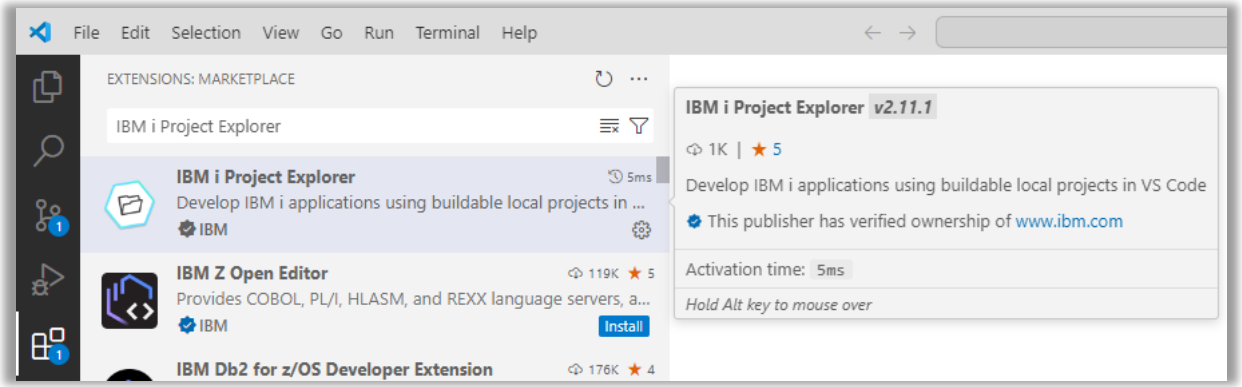
Installation

IBM i
continuous innovation
continuous integration

1 *Download*
Visual Studio Code



2 *Download VS Code extensions*
IBM i Project Explorer,
Source Orbit and *Code for IBM i*
or just the IBM i Development Pack

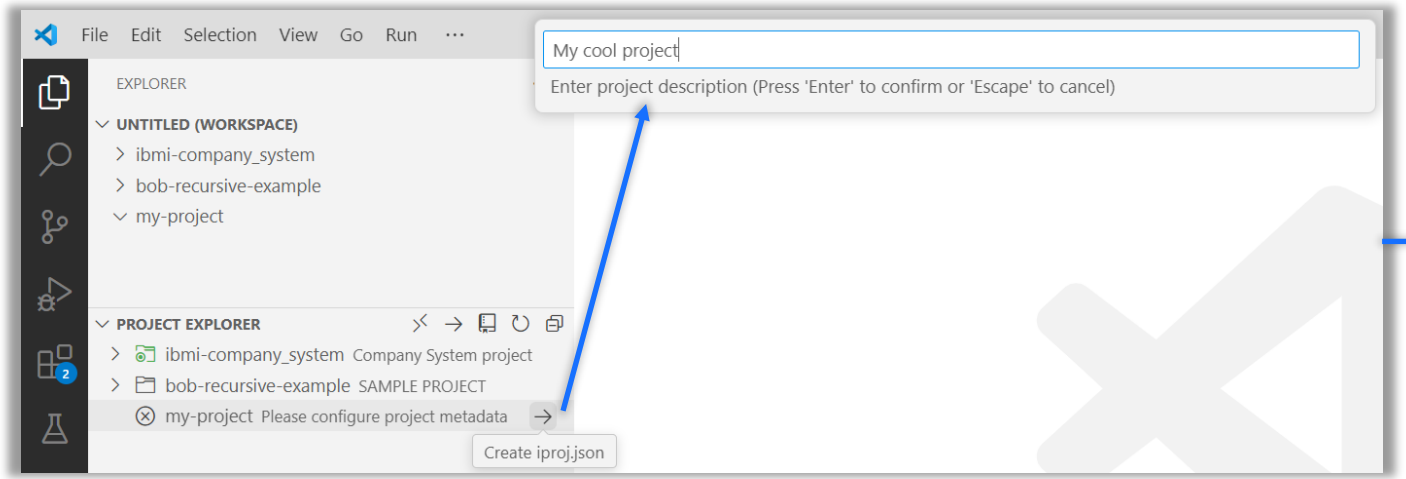


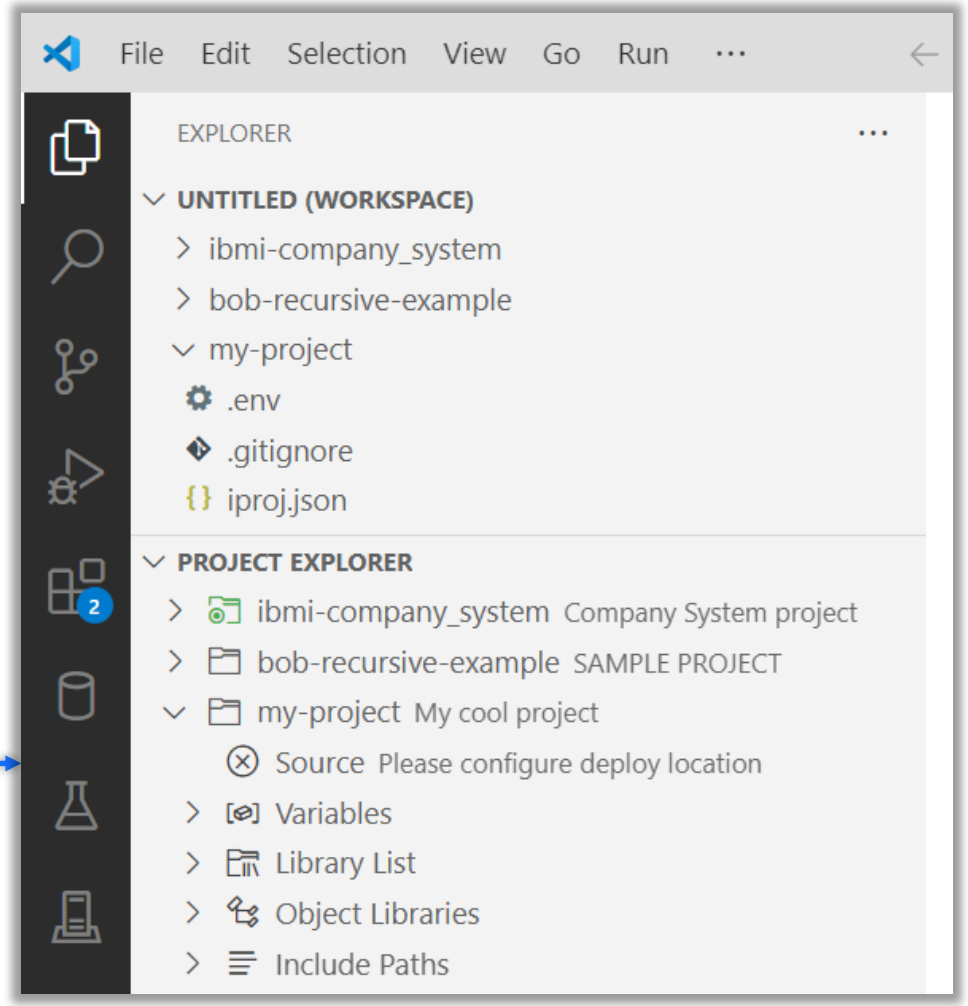
3 *Run*
yum install bob
on IBM i

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

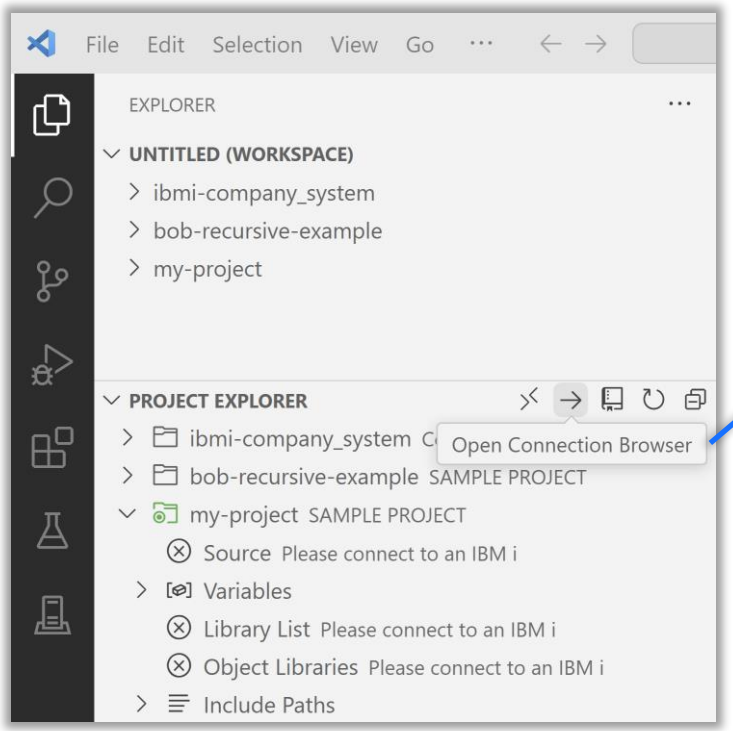


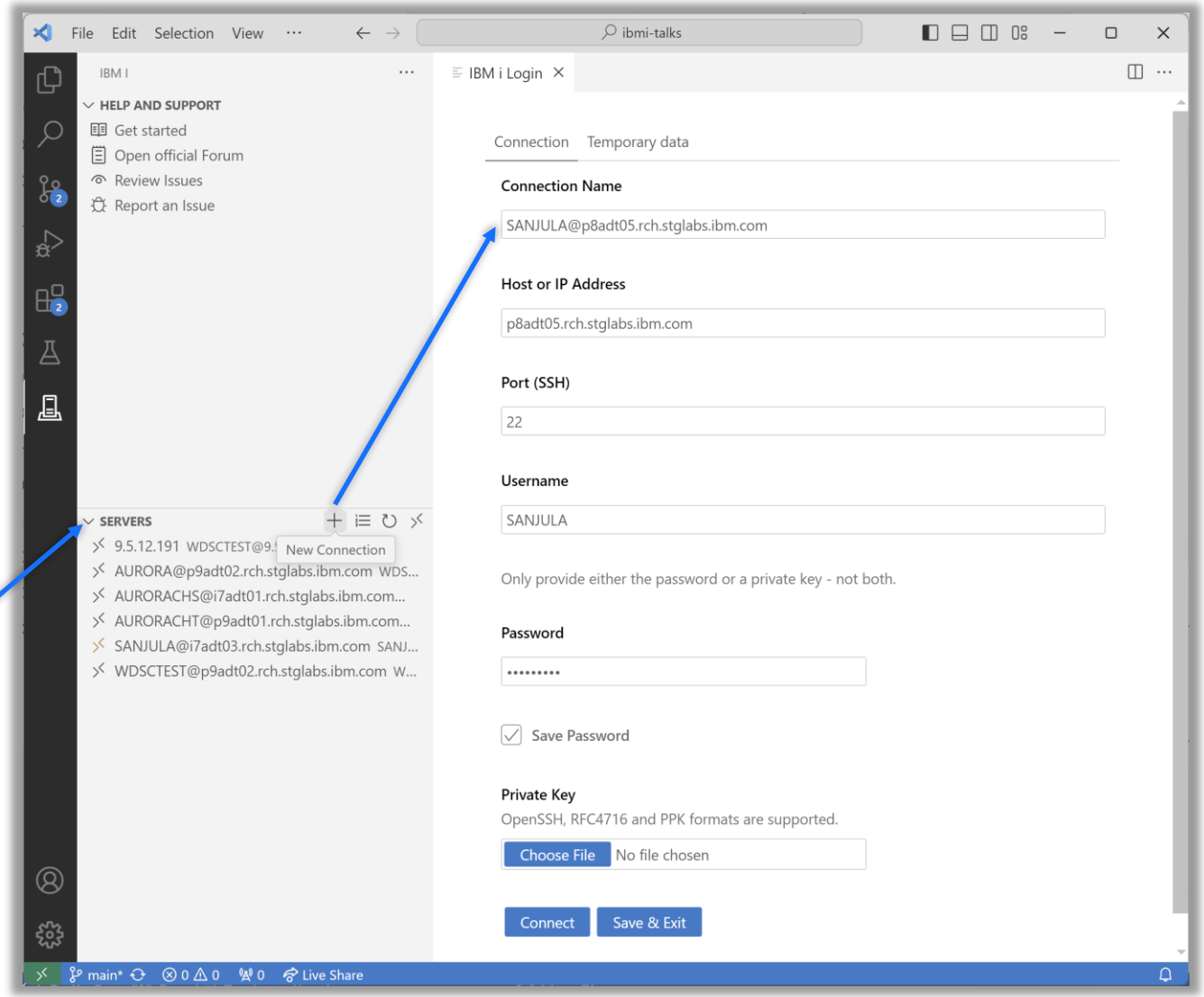


Connect to an IBM i

IBM i
continuous innovation
continuous integration

- Open the *Connection Browser* from Project Explorer
- Create new IBM i connection from the *Server* view





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

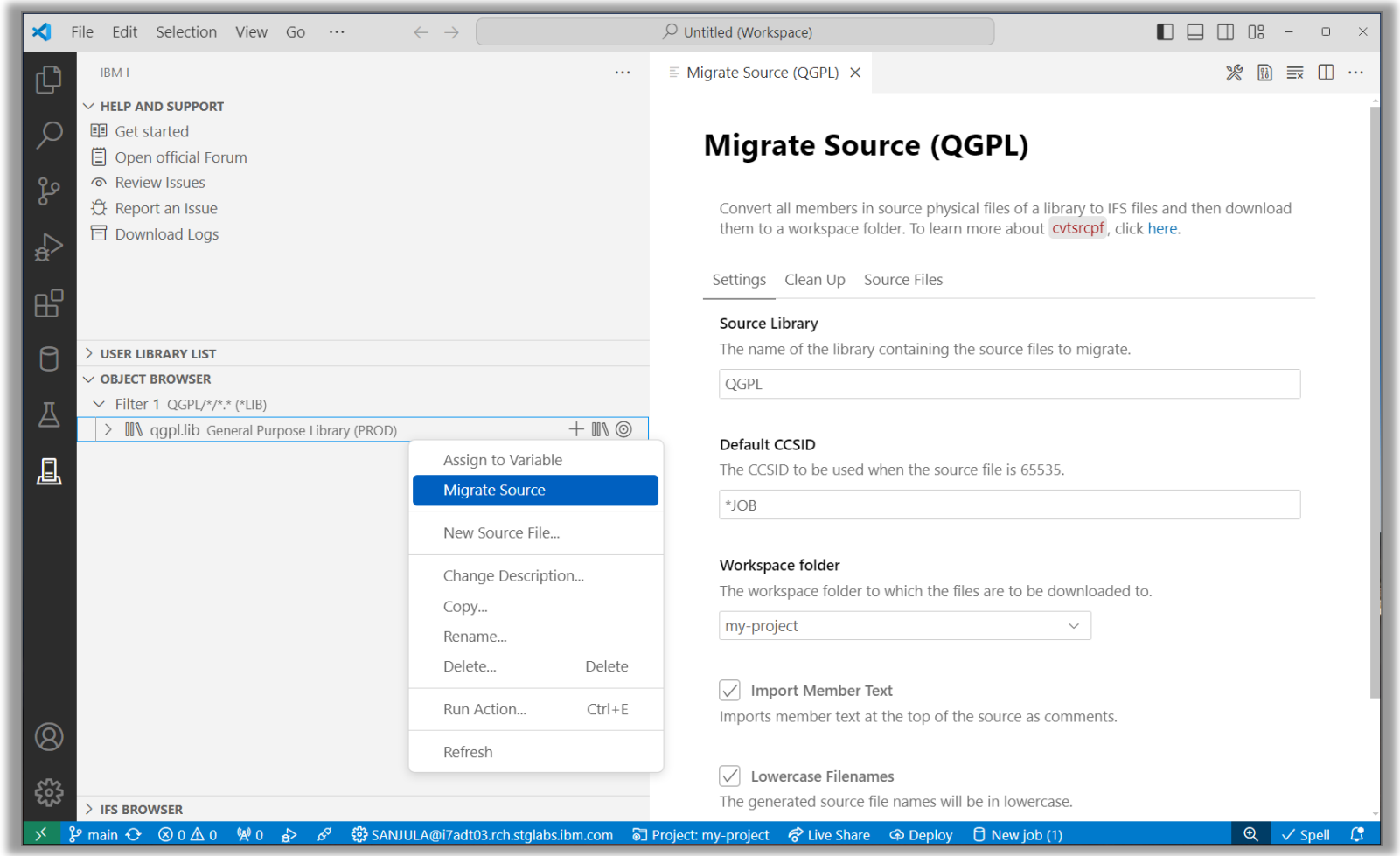
↓

Source Orbit

Rename extensions

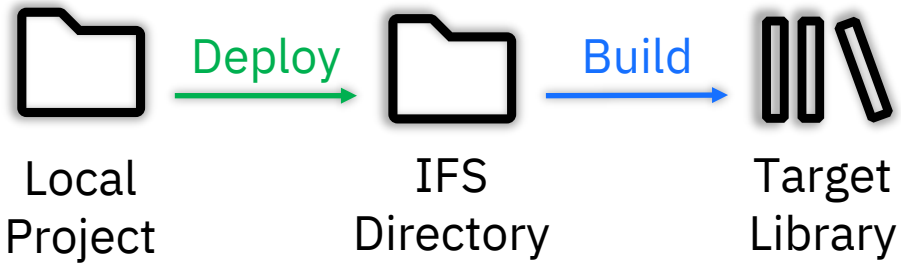
↓

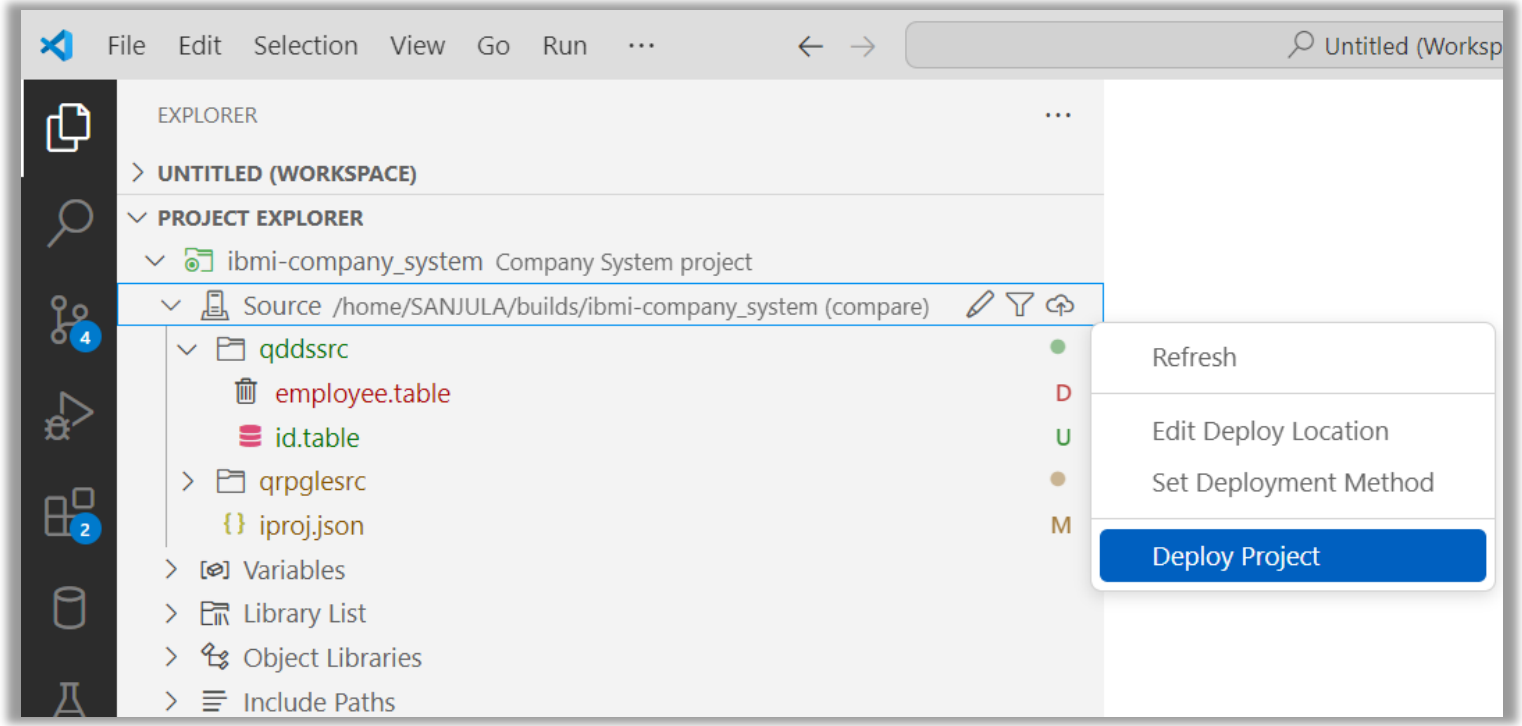
Convert includes/copy directives to Unix style paths



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 (typically the fastest)
  - Working Changes
  - Staged Changes
  - All
- Deploy project
  - Moves files to deploy location based on deployment method

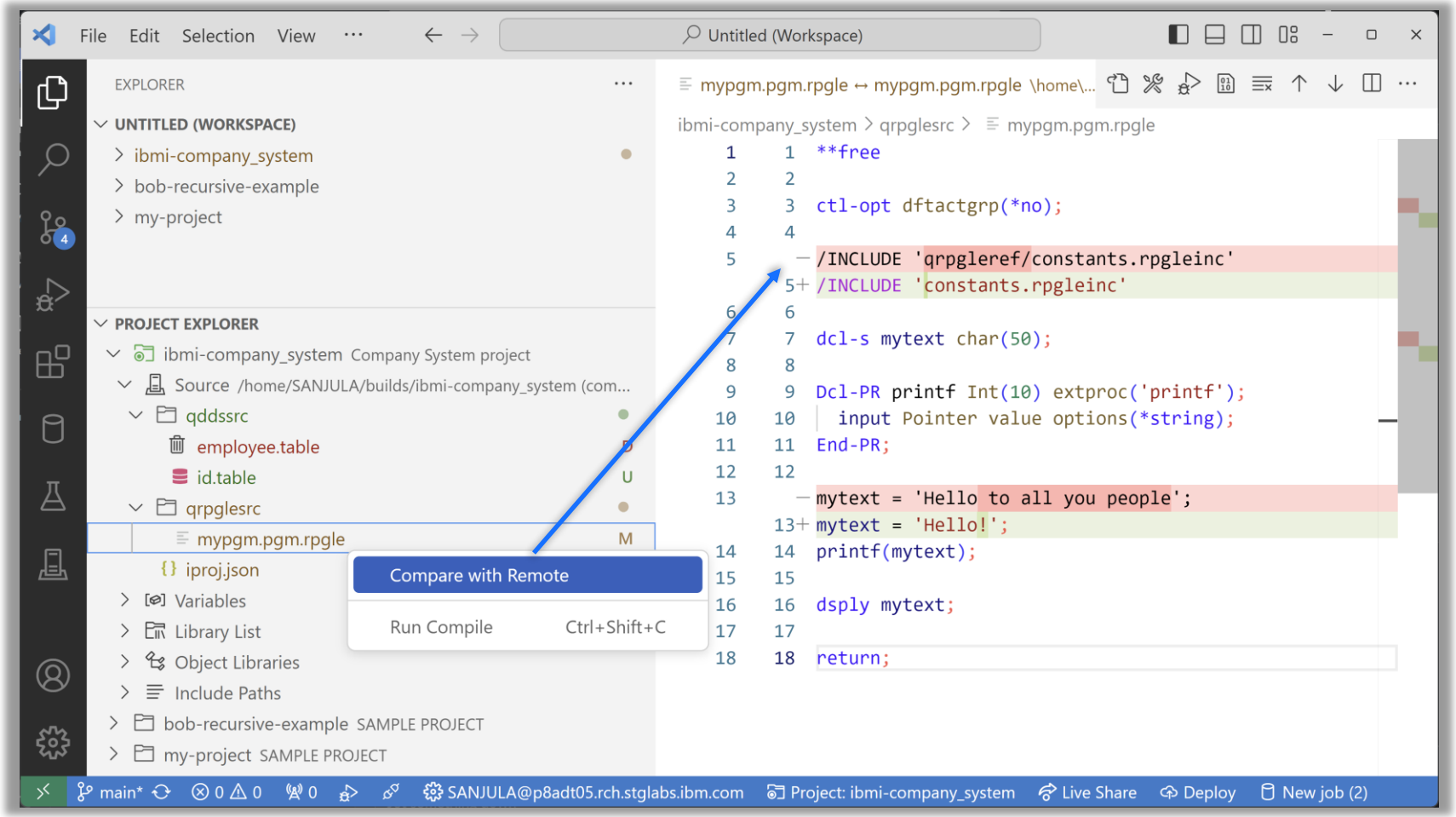




Visualize Local vs. Remote Source Files

IBM i
continuous innovation
continuous integration

- Visualize, compare, and deploy your local source files to the deploy location in the IFS
- Track file changes (added, modified, deleted, etc.)
- Compare local file content with remote IFS

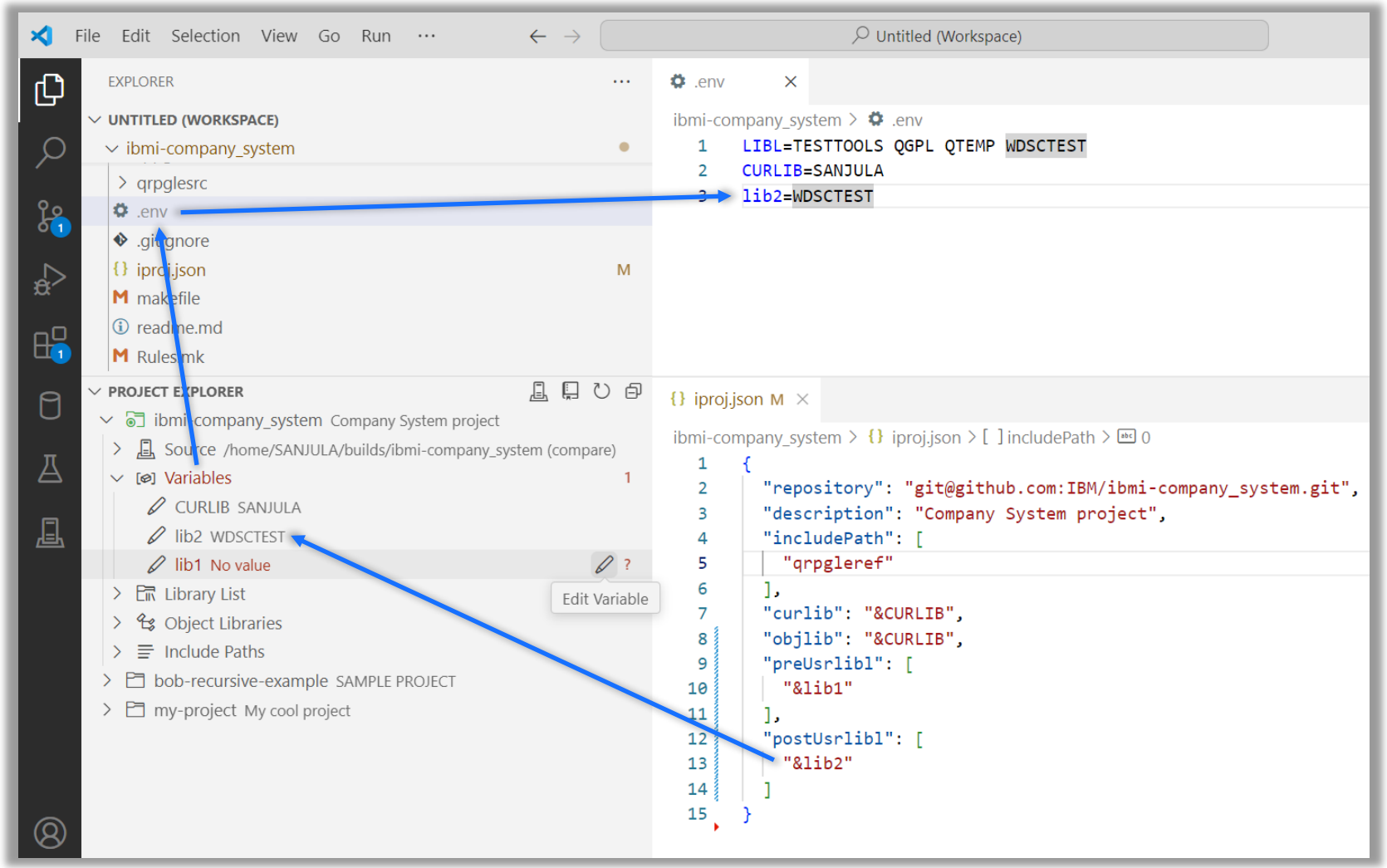


Work with Variables

IBM i
continuous innovation
continuous integration

- Reusable project definition that can be used by multiple developers or in automated builds
- 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

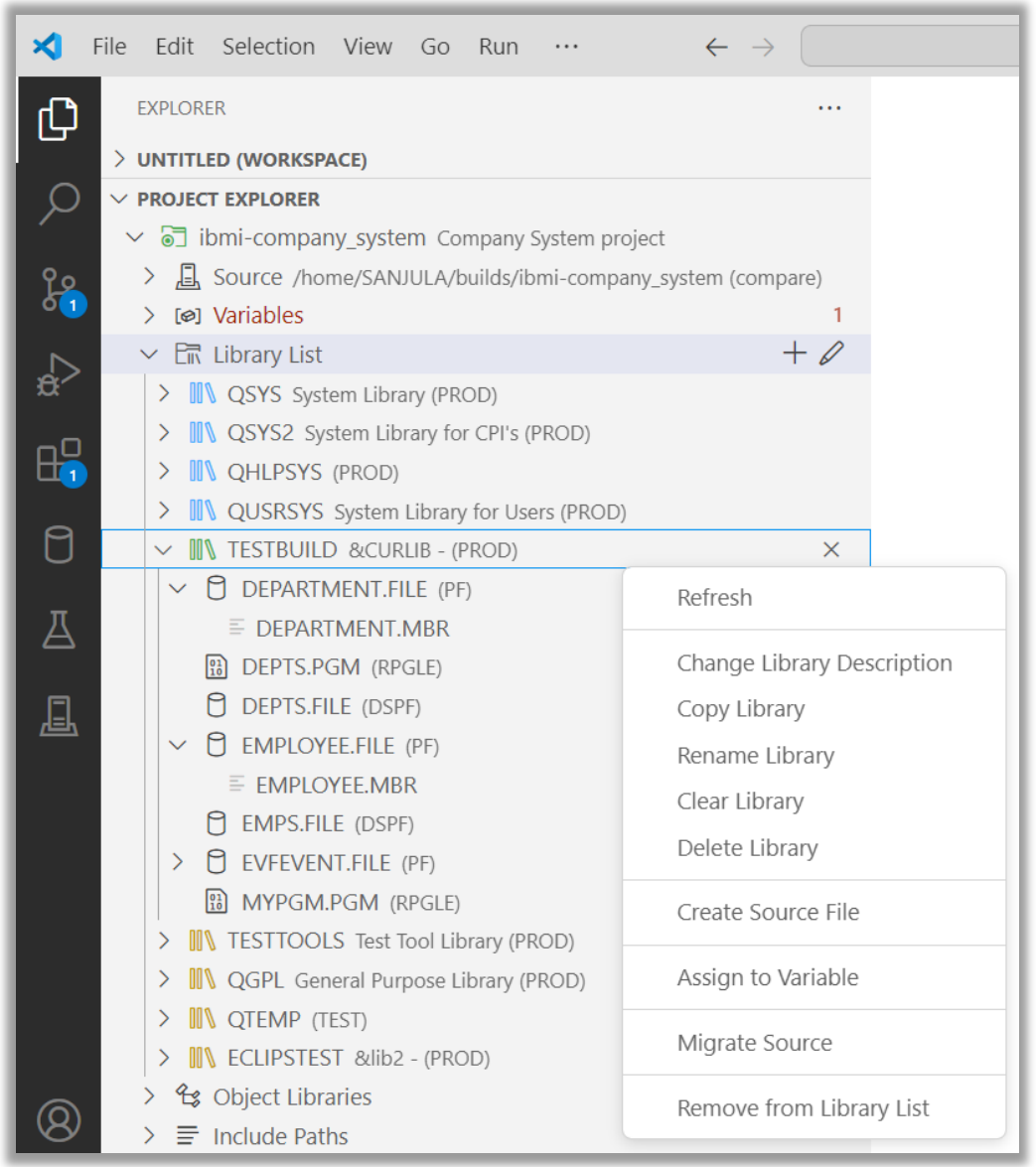
Do not push .env file to Git!



Manage the Library List

IBM i
continuous innovation
continuous integration

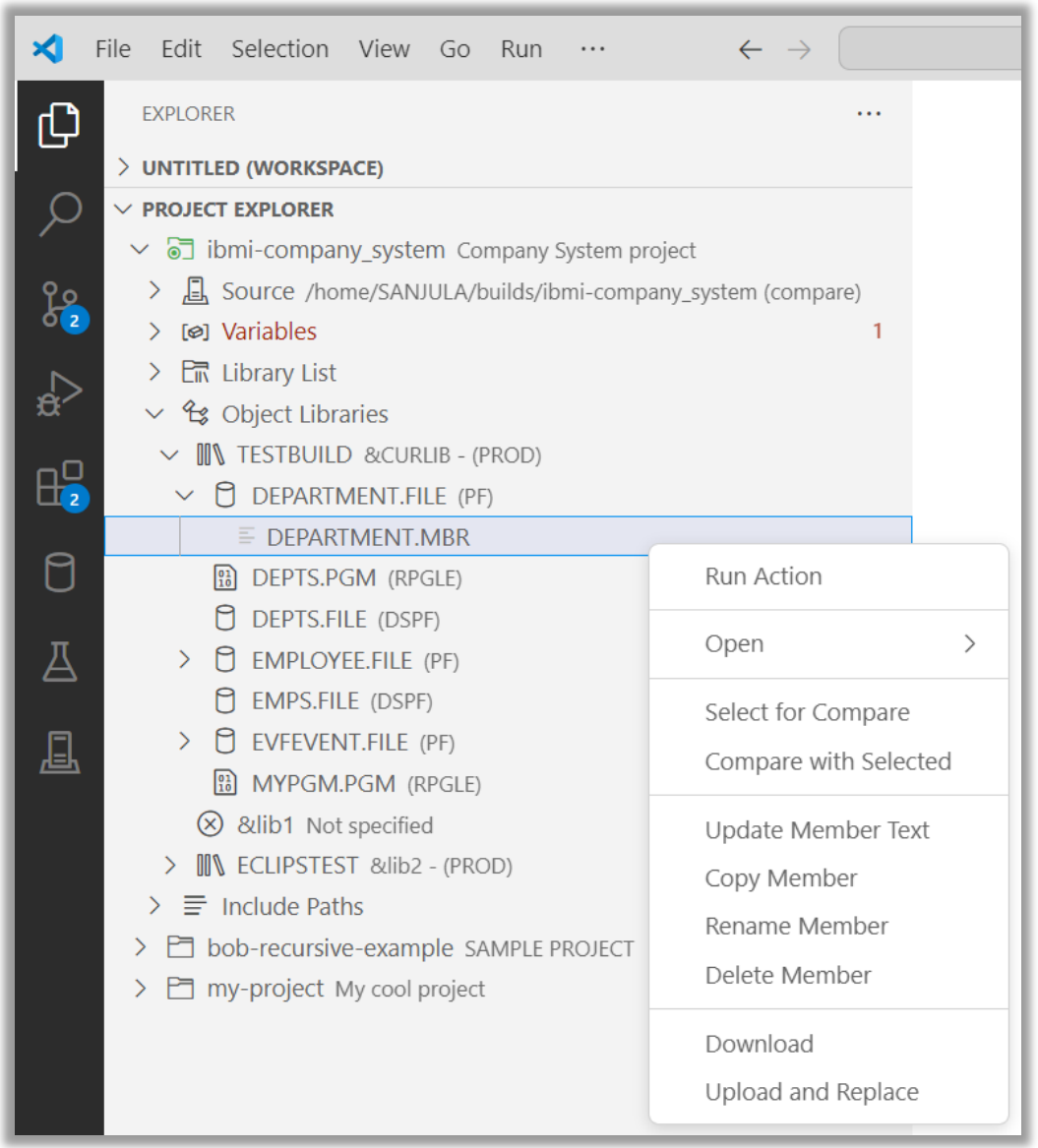
- Project's library list is a composition of your user profile's library list (from JOBD) + set of project specific libraries
- Add to beginning/end of library list (*preUsrlibl* and *postUsrlibl*) and set current library (*curlib* in iproj.json)
- Reorder library list
- Browse objects and members
- Manage libraries, objects, and members



Browse Object Libraries

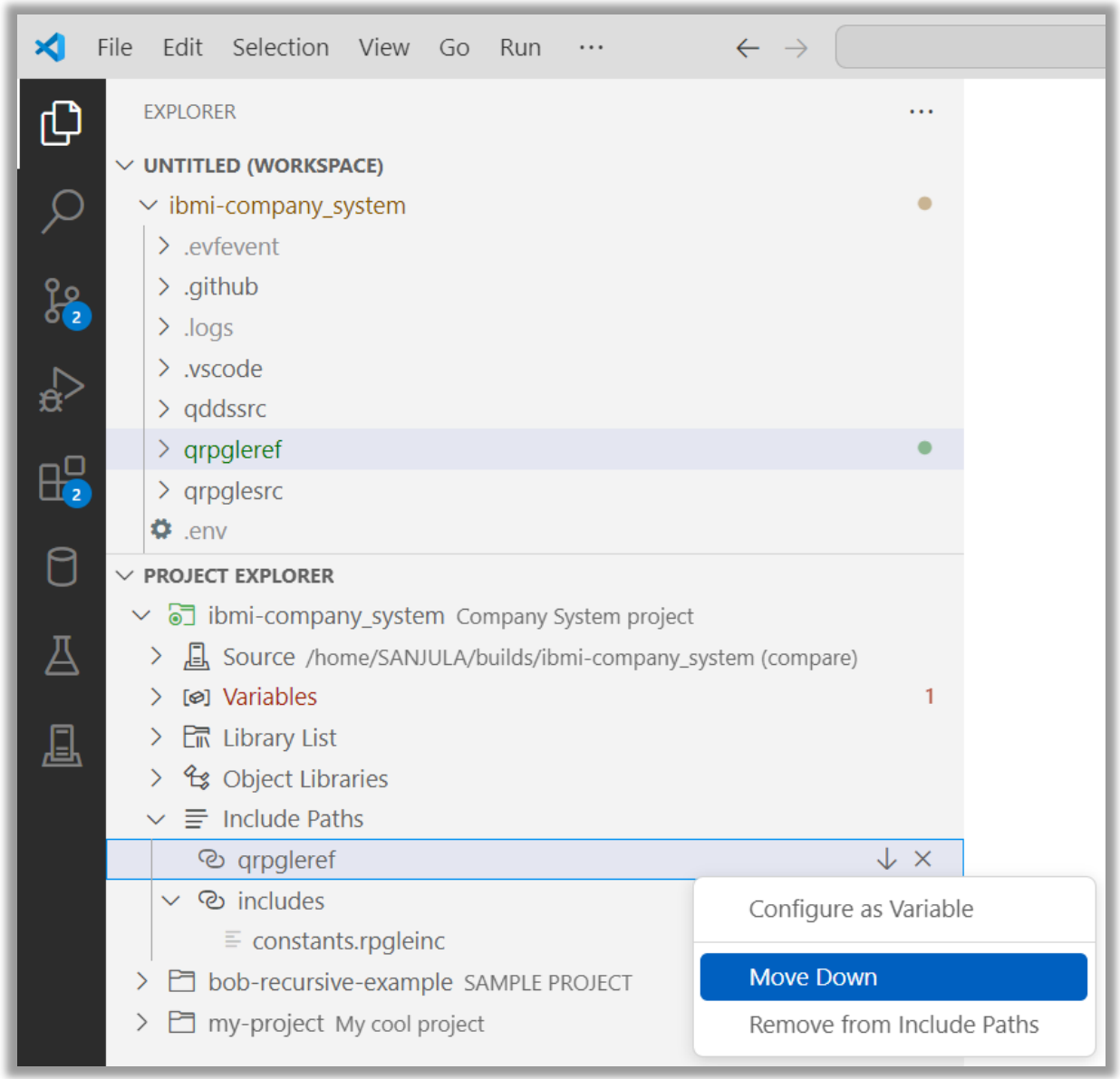
IBM i
continuous innovation
continuous integration

- The place for developers to easily see, debug, and manipulate the results of your build
- Another place to manage libraries in iproj.json (*curlib*, *objlib*, *preUsrlibl*, *postUsrlibl*)
- Manage libraries, objects, and members



Manage Include Paths

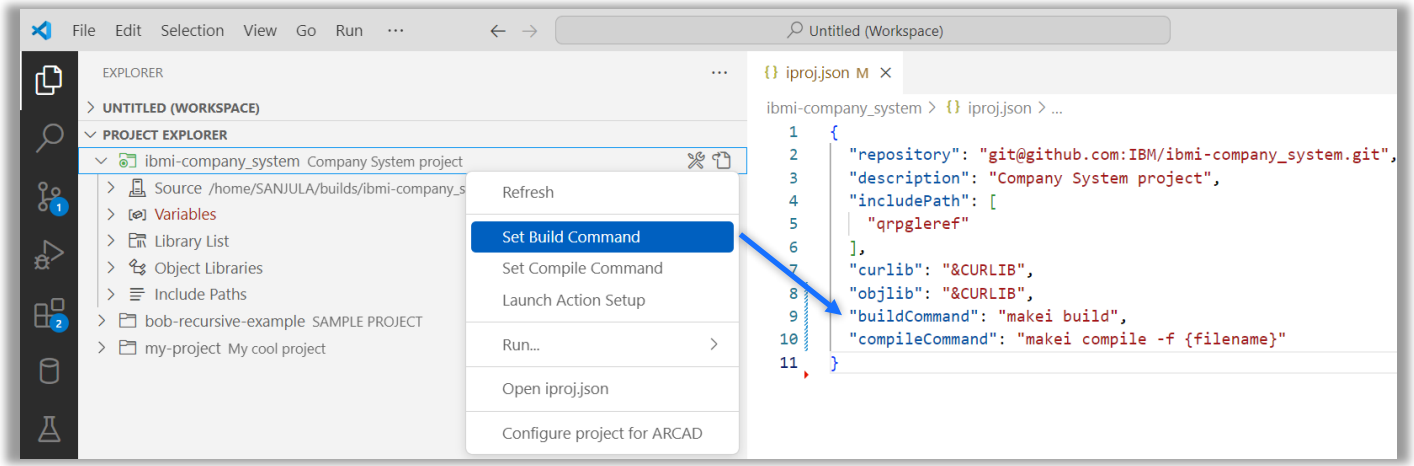
- Self-contained projects should know where to find includes within the project
- Add, remove, and reorder include paths
- Visualize if includes resolve locally or to remote IFS



Build and Compile

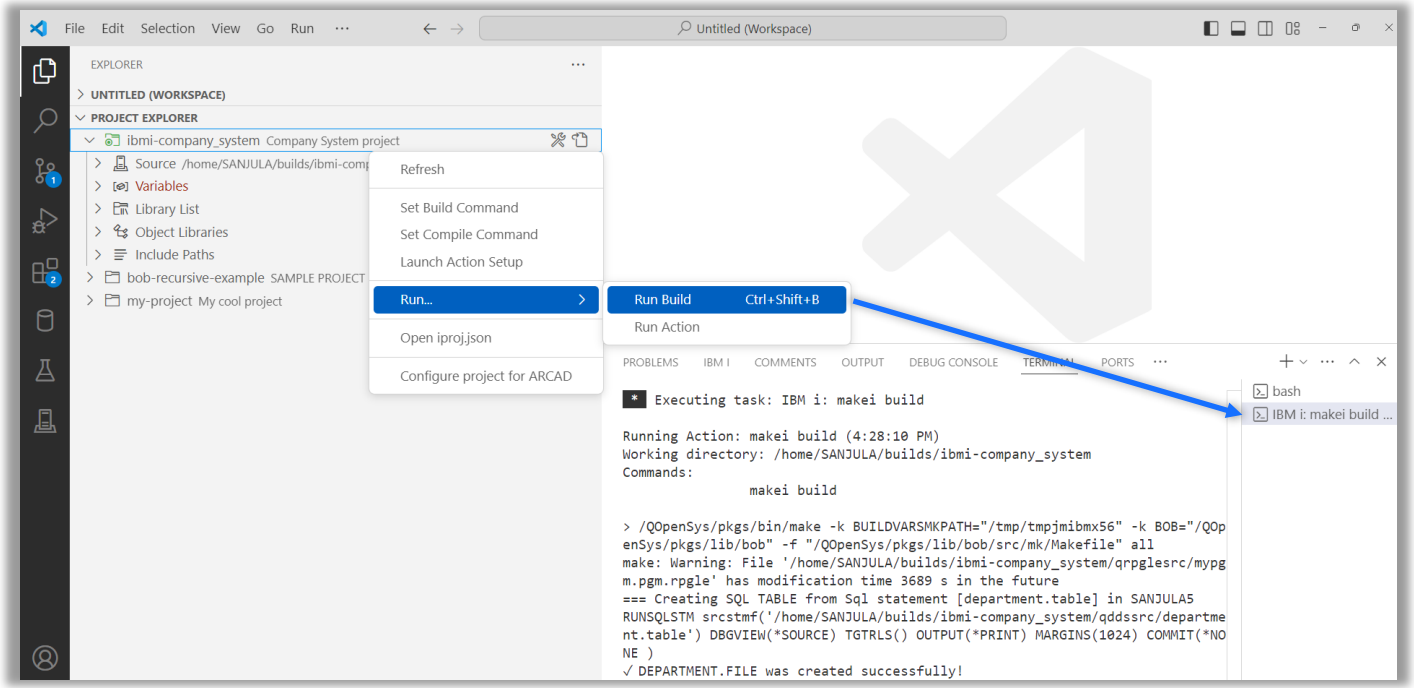
IBM i
continuous innovation
continuous integration

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



- Building
  - Set build command
  - Run Build (*Ctrl+Shift+b* or *Cmd+Shift+b*)

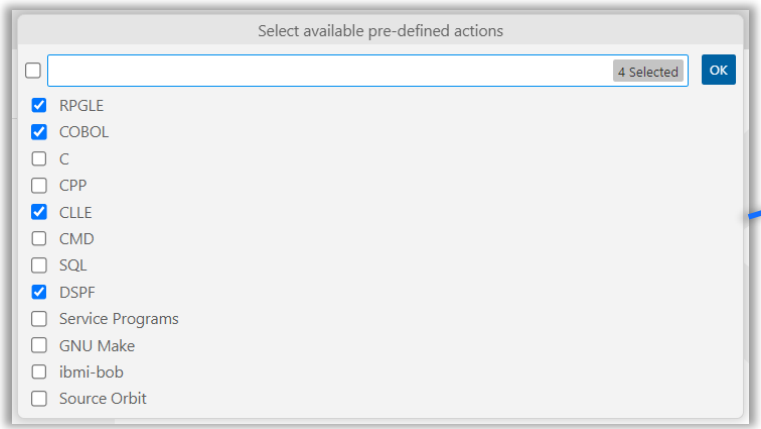
- Compiling
  - Set compile command
  - Run compile (*Ctrl+Shift+c* or *Cmd+Shift+c*)
    - On active editor
    - On file or directory in File Explorer
    - On file or directory in Source

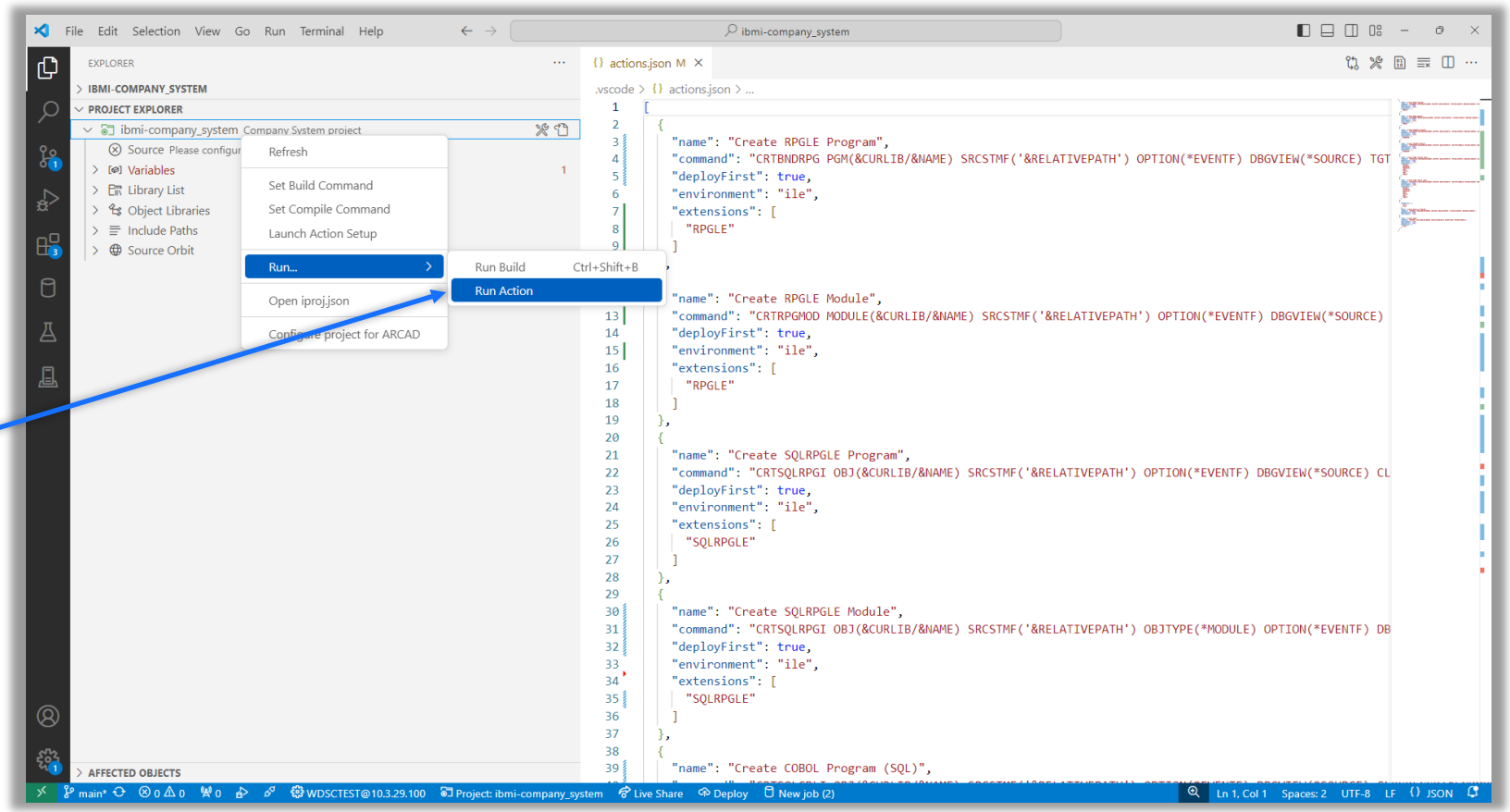


Run Actions

IBM i
continuous innovation
continuous integration

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

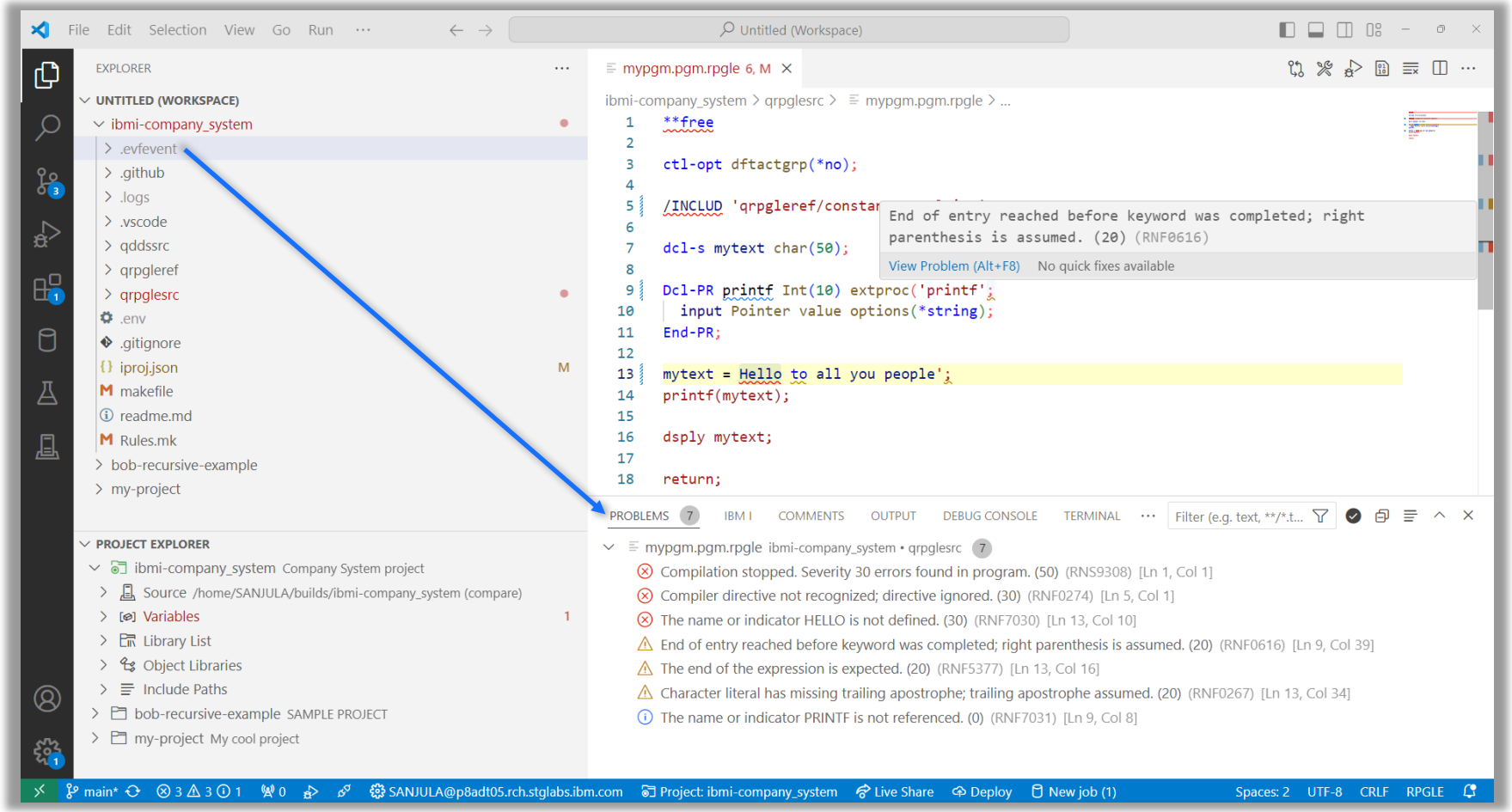




View Diagnostics

IBM i
continuous innovation
continuous integration

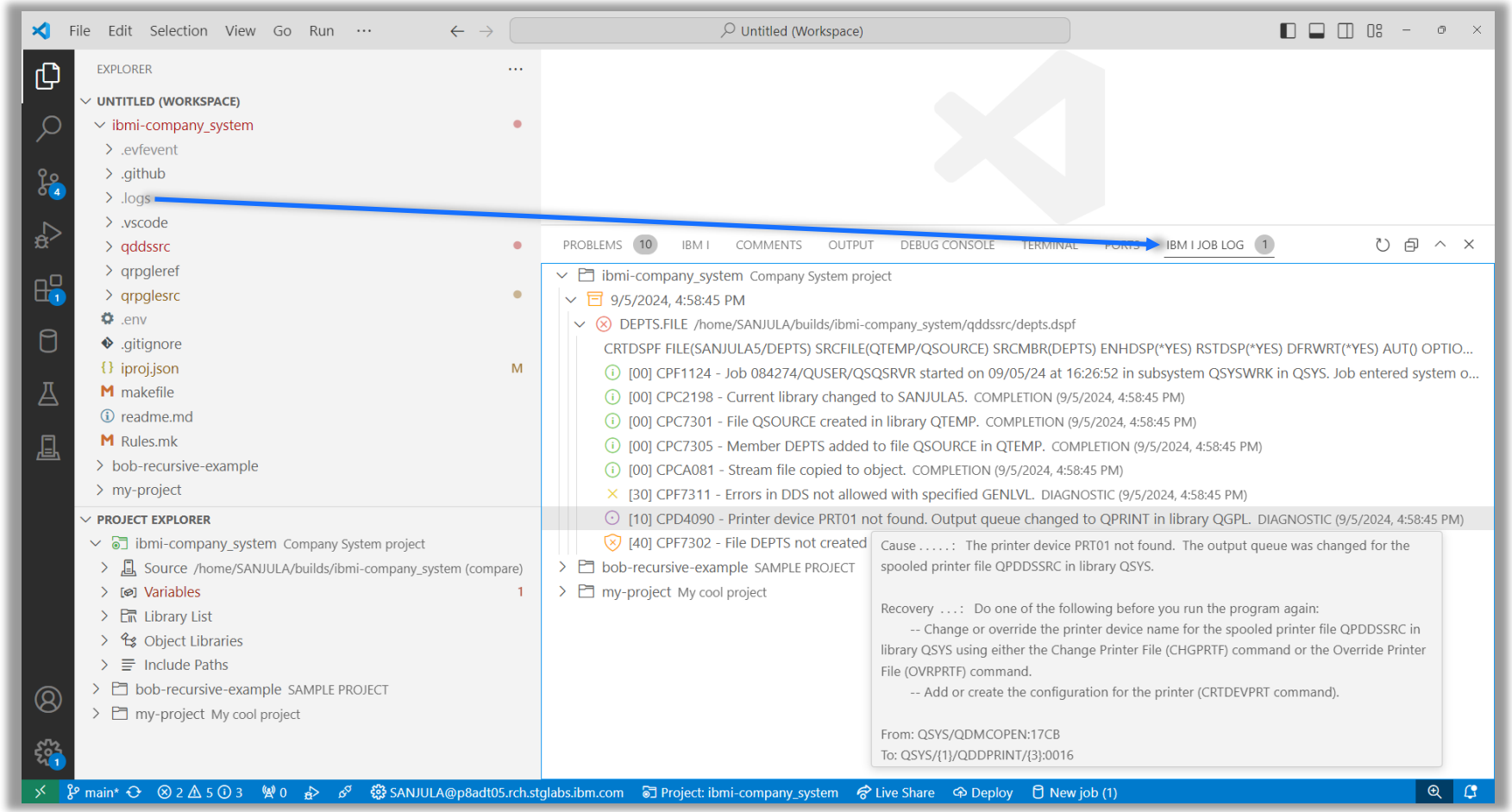
- Visualize build or compile diagnostics in the Problems view
- Evfevent file diagnostics are dumped in a .evfevent directory after a build or compile
- Diagnostics are also rendered inline in the source file



View Job Logs

IBM i
continuous innovation
continuous integration

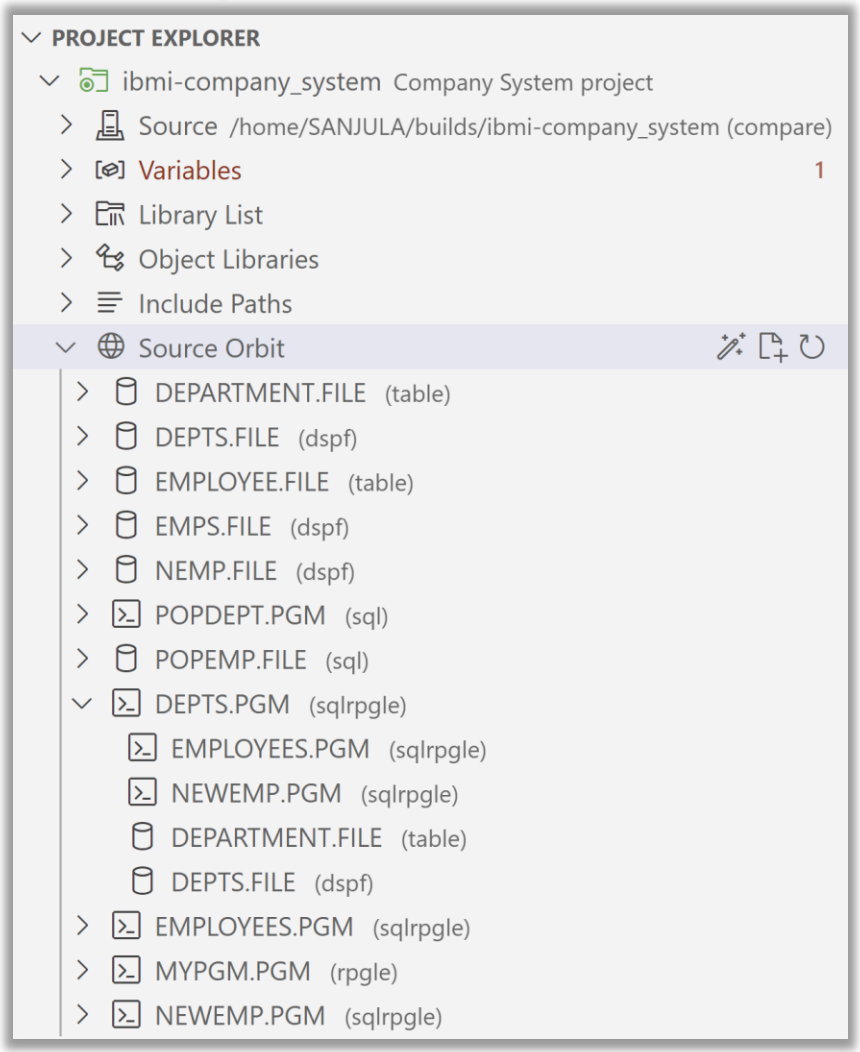
- Visualize and manage anything that could be seen in an IBM i job log including second level help
- Job log and spool files are dumped in .logs directory after a build or compile
- Track up to 10 of the previous logs in memory
- Organized by the ILE objects being built
- Filter by failed objects or severity



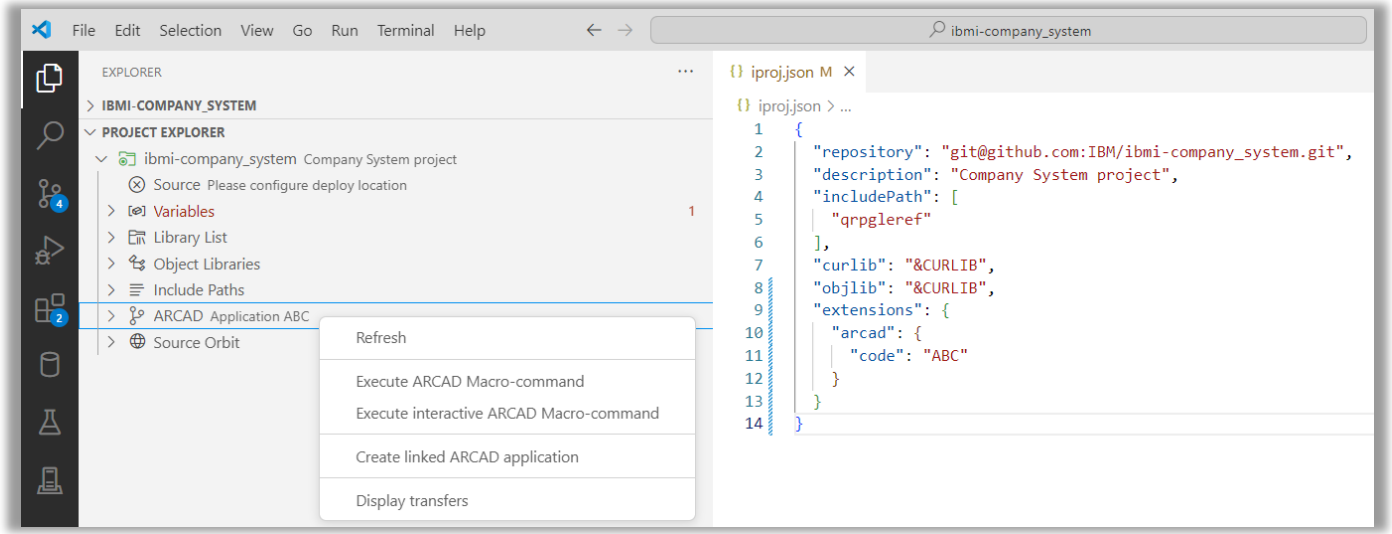
Integration

IBM i
continuous innovation
continuous integration

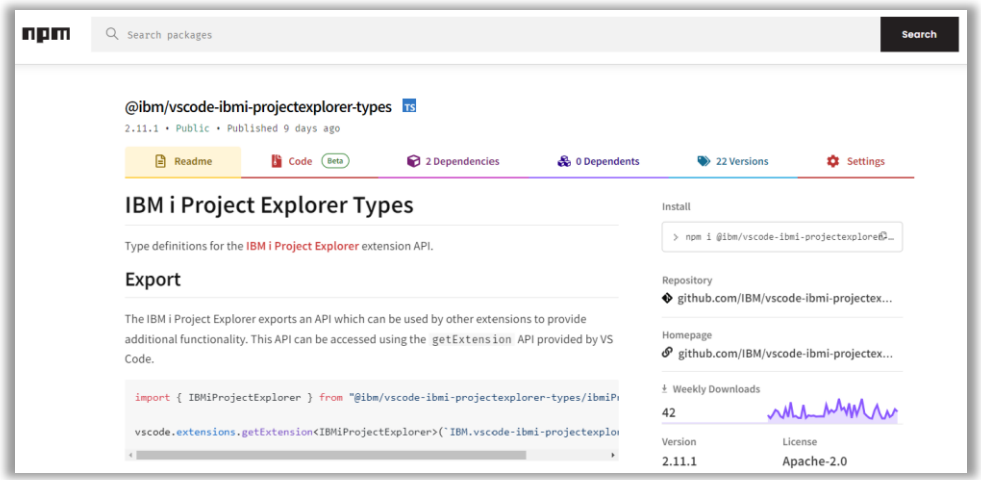
Source Orbit



ARCAD-Elias



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



Any Questions?

Important 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

IBM