

Intro to Git Source Versioning of Classic IBM i Source with iForGit

For IBM i Developers

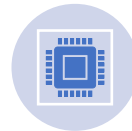


Your Presenter



Richard Schoen

richard@richardschoen.net
richard@mobigogo.net
richard@seidengroup.com



30+ yr developer
IBM i, Windows,
Linux, Mac



Started RJS
Software Systems
in 1990



Created automated
report distribution,
doc management
and app integration
software



Sold business in
2014



Started MobiGoGo
to build multi
platform apps



Provides training and
consulting services



Contributes open
source to Github
repositories



Author of **iForGit Native IBM i Git client**
for RDI, SEU and PDM developers

Github: <http://github.com/richardschoen>

Business site: <http://www.mobigogo.net>

Personal site: <http://www.richardschoen.net>

LinkedIn: <https://www.linkedin.com/in/richardschoen>

Twitter: @richardschoen

Blog: <http://blog.richardschoen.net>

iForgit IBM i Git Client: <http://www.iforgit.com>



Discussion Topics

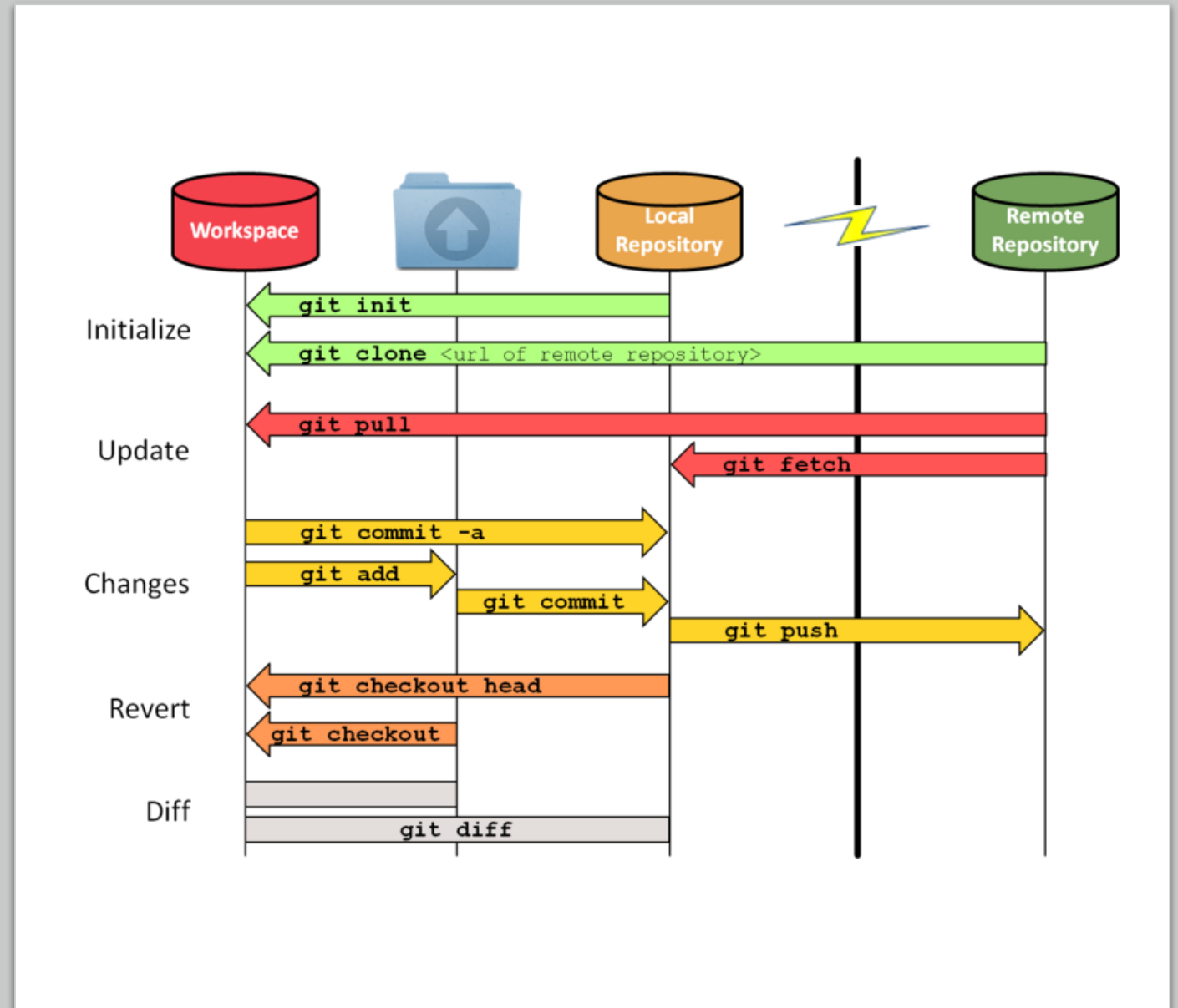
- Managing IBM i Source Physical Files with Git
- Intro to iForGit Git Client for IBM i
- Demo

What is Git ?

- Git is an open-source distributed **version control** software component written by **Linus Torvalds**, the creator of the **Linux** kernel.
- The program was designed to keep track of every change that occurs in a specified PC or IFS style directory.
- **GitHub** is a web-based Git repository hosting service.
- GitHub Slogan - “**Build better software together**”
Acquired by Microsoft in 2018
- **Git knows Nothing about IBM i Source Members in Source Files**

What is a Git Repository (Repo)?

- Stores and saves project file changes
- Tracks file changes
- Knows who changed a file
- Can be created locally on PC or IBM i (IFS)
- Can be pushed remotely to GitHub, GitLab, Azure, etc.



How is Git different from other IBM i Source Tools

- Git does not manage your objects/builds. It is source version control only.
- Git does not automatically provide build processes
- By itself is not an IBM i source file management tool
- Does not provide issue, wikis, etc. without something like GitHub
- There are other commercial IBM i source tools that manage source/objects

Two types of IBM i Git Projects

- IFS or PC based source
- Source Members in source physical files



IFS or PC Based Projects

- IFS based source
- Node, PHP, Java, Python, etc.
- App contains multiple file assets
- html, css, js, php, python, etc
- Edit many source members simultaneously
- App usually builds at runtime
- Commit many source member changes to git at once



IFS or PC App
/application
HTML
CSS - .css
Javascript - .js
PHP - .php
Python - .py
NodeJS - .js

Classic IBM i Source Physical File Projects

- All Source Members in source files
- QRPGLSRC, QCLSRC, QDDSSRC, etc.
- Edit one source member at a time
- Still edit from source files as always
- Source member locking for concurrency
- Git adds source member change visibility
- Source physical file members committed to git repository



IBM i Library
GITTESTxxx
QCLSRC
QCMDSRC
QDDSSRC
QRPGLSRC
QSQLSRC
QTXSRC

Summary of Basic Git WorkFlow

For PC or IFS Files



- Create new local git repository – **git clone url://mygitrepo.git** or **git init** .
git supports many url types: ftp, http, ssh, .
- Clone existing git repository from GitHub, Bitbucket, GitLab, etc.
- Edit files in working directory
- Add changed files to staging area – **git add foo.py bar.js baz.rb**
- Commit staging area to repository – **git commit -m 'Commit message'**
- (Optional) Push history to remote repository – **git push**
- (Optional) Pull history from remote repository – **git pull**

What does GitHub Provide ?

- GitHub Plans
<https://github.com/settings/billing/plans>
- Private documentation Pages and Wikis
- Repository insights
- Multiple issue assignees
- Required reviews
- Required status checks
- FREE plan may be good enough to start managing source code


Creating Your GitHub Repository

Or Use GitLab, Bitbucket, Azure DevOps, Gitbucket etc.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner *

 richardschoen ▾

/


Repository name *

demo01 ✓


Great repository names are short and memorable. Need inspiration? How about [verbose-barnacle?](#)

Description (optional)

My Demo Repository

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☒ **Add a README file**

This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**

Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**

A license tells others what they can and can't do with your code. [Learn more.](#)

\$ **git init** (Creates a bare repository in your current directory)

\$ **git clone** <https://github.com/jordiwes/demo01.git> (Clones an existing repository in your current directory)

What is a Git Client ?

- Git clients are tools that allow developers to access their git repositories
- May provide a graphical user interface (Tortoise, SourceTree)
- IDE's may have one. Visual Studio, VS Code, Eclipse (Egit)
- Default client is PC, Linux, Mac or IBM i PASE based command line calls
- iForGit is a 5250 based CL driven git client for PDM, RDI and SEU users
- iForGit knows how to manage IBM i source members with git

Lots of PC Git Client Choices

- Git Install
<https://git-scm.com/downloads>
<https://git-for-windows.github.io>
- Command line (Windows, Linux, Mac, IBM i - bash)
- Windows Shell Interface to Git
<https://tortoisegit.org/>
- Git GUI Clients
<https://git-scm.com/downloads/guis>
- SourceTree
<https://www.sourcetreeapp.com>
- Google: **git clients**

What is iForGit ?

Simplified IBM i Git Workflows

- Git Client for PDM – (Program Development Manager)
- Git Client for SEU – (Source Entry Utility)
- Git Client for RDI (Rational Developer for i)
- Keep source member in source physical files
- Driven by CL commands
- Schedule daily changed member commits
- On demand member commits (PDM or RDI)
- Automatic commits with SEU (If desired)
- Auto-store source versions in the IFS or push Git repository to Github or other online repository
- Changes viewed online at Github or via Tortoise, SourceTree, etc.



What iForGit Is Not – Today

An Object Management or Build Tool

- A full devops lifecycle toolset
- An automated continuous integration build tool (CI)
- A visual git client for viewing source versions
Use Tortoise, SourceTree, etc.
- **Not a good vehicle for using git branching - yet**
- **1:1 each library is tied to a single git repository**
- **Isn't required for IFS PHP, NodeJS or other IFS development**
- Although there are general CL commands to drive all git operations



Why Choose iForGit

- **Fast and easy** classic IBM i source file versioning
- Need to get a handle around source changes and versioning
- Be able to view source changes (via git repo)
- **Automatic** source versioning
- **Compliance** issues with auditors
- **Installs in minutes**
- Don't need large source/lifecycle management tool
- May not need automated build tooling
- May want to use Github or other best of breed project management and build solutions



No Initial Changes to Development Process

- Zero initial learning curve for developers
- Edit source file members as always
- Use existing IBM i member locking
- Back up libraries nightly/weekly as usual
- Adds source member change visibility and auditability with Git

Reduce Source Member Version Copy Hell



- Example: Need to change SRC001R
- Copy to: SRC001RS1, SRC001RS2, etc.
- Try to use member text to describe differences
- Lose track of previous member version sequences
- No way to visually compare changes between member versions

iForGit Main Source Versioning Use Case



- Instant IBM i classic source member version control
- Continue to keep source in standard source files
- Use traditional source editing with SEU or RDI
- Low or NO learning curve for existing developers
- Automatically version and back up source changes to a Git repository
- Keep repo in IFS or store remote Git versions on Github, Gitlab, etc.
- Eliminate source member copying
- Ease in to using Git



iForGit Automated Passive Versioning Use Case



- Instant version control and automatic scheduled change capture
- Continue to keep source in standard source files
- Use traditional source editing with SEU or RDI
- No learning curve for existing developers
- Automatically version and back up source changes to a Git repository
- Keep repo in IFS or store remote Git versions on Github, Gitlab, etc.
- Eliminate source member copying
- Add change visibility
- Ease in to using Git



iForGit Edit, Commit, Push Changes Use Case



- Developer driven change commits
- Can be used with passive change capture in place
- Commit latest version before editing source
- Commit after editing source
- Eliminate source member copying



iForGit Augment Custom Build System Use Case



- Replace source control for existing custom build system
- Integrate source member commit, push and pull from repo
- Use CL commands from your custom build process



IBM i Source Member Management Assumptions

- You will still be editing, compiling and building as usual
- There will be 1 shared git repository for each library
- Your git repo will mimic your library/file/member structure in the IFS
Ex: `/gitrepos/qgpl/qclsrc/qstrup.clp`
- One-way version push from library to git repository
- Can view, copy and paste source back from git repo
- At some point you want to create your own custom IBM i build process
- At some point you want to create your own custom IBM i deploy process

IBM i Source Repository IFS Structure

- Think of each library as an IFS git repository/project directory
- The git repository for each library gets created at the library level
- Can create top level directory for all git repos or just each library (library by default)

Library name: **QGPL**

/gitrepos/qgpl – Top level plus library level for git project repository

/qgpl – Library level directory

- Each source file gets a dir under library directory

/gitrepos/qgpl/qclsrc

“

“/qrpglesrc

“

“/qddssrc

- Under each source file subdirectory, store individual source member files

/gitrepos/qgpl/qclsrc/pgm001.clp

“

“/qrpglesrc /pgm002.rpgle

“

“/qddssrc /customers.pf

Source File Master in Libraries

Git Local Library Repository in IFS



IBM i Library

GITTESTxxx
QCLSRC
QCMDSRC
QDDSSRC
QRPGLSRC
QSQLSRC
QXTSRC

No change to existing
SEU or RDI editing
process needed initially

One Library Per Git Project Repository

IFS Git Repository
/gitrepos/GITTESTxxx
QCLSRC
MEMBER01.CLP
QCMDSRC
MEMBER01.CMD
QRPGLSRC
MEMBER01.RPGLE

Local IBM i Version of
Source in IFS Directories

View /Commit
Changes with
Git Clients:
Tortoise Git
Source Tree

<\\server\gitrepos\GITTESTxxx>
IFS Share required to view from
IFS Git repository

View git repository
vis IFS file shares

Source File Master in Libraries

Git Repo in IFS and GitHub/Remote Repo



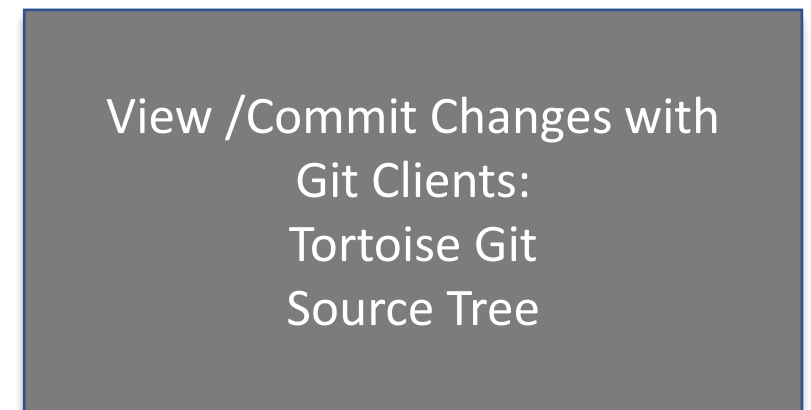
No change to existing
SEU or RDI editing
process needed initially



Local IBM i Version of
Source in IFS Directories



Push,
replicate
changes to
Remote git
repo



[\\server\gitrepos\GITTESTxxx](#)
IFS Share required unless using
remote Git repository

Library Git Versioning Strategy

- Send main source library source members to library git repo
- Send developer/work libraries to git repo for work in process
- Stop copying/pasting source members to new member for new versions
- Simply commit current source member version before/after changes
- If copying/pasting source to dev libraries and back, things to consider.

Main Source and Dev Source Copies

- Copy source from main library to your dev/work library
- Edit, compile, test in your library and version to git while work in process
- Run commit on main library source version to make sure commits current
- Copy source back from dev/work library to main source library
- Run commit again main library source version to capture dev/work changes
- Compile and deploy as usual into production

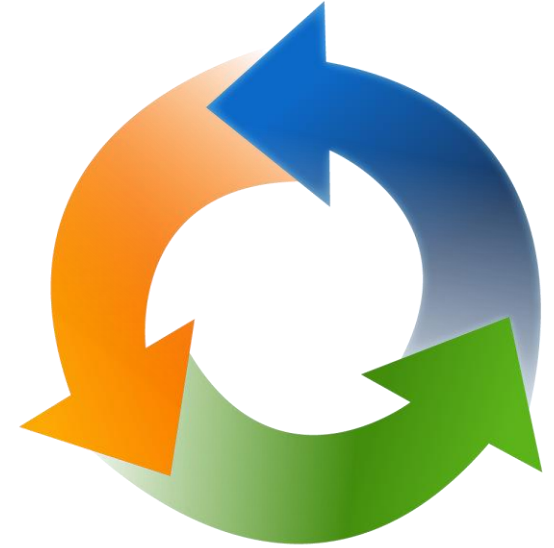
iForGit Basic IBM i Source Member Git Flow



- Create local git IFS repository – git init
- Run LIBSRCEXP *ALL to do initial git population from a library
- Edit source members as usual
- Run SRCTOGIT command or use daily auto-commits via LIBSRCEXP
Add changed files to staging area – git add filename.rpg, etc.
- Commit staging area to repository – git commit -m 'Commit message'
- (Optional) Push history to remote repository – git push
- (Optional) Pull history from remote repository – git pull

Classic IBM i Development

- Edit Source with SEU or RDI
- Edit, Save
- Compile
- Test
- Repeat...
- Deploy to Production
- Back Up Source Libraries



Classic SEU/PDM IBM i App Development



IBM i Source Editing Process



```
Columns . . . : 6 76      Edit      GITTEST123/QRPGLESRC
SEU==>      MONOSTDOUT
FMT *      *. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+...
***** Beginning of data *****
0001.00 *-----
0002.00 * Source information
0003.00 * @@LIBRARY: GITTEST123
0004.00 * @@FILE: QRPGLESRC
0005.00 * @@MEMBER: MONOSTDOUT
0006.00 * @@TYPE: RPGLE
0007.00 * @@TEXT: Read and parse stdout log....xxz
0008.00 *-----
0009.00 /*****
0010.00 * Program: MONOSTDOUT
0011.00 * Author : Richard J Schoen
0012.00 * Desc . : This program will read the selected STDOUT log file and
0013.00 *          send back info to the caller for processing the
0014.00 *          selected log messages. MONOSTDOUT is created in QTEMP
0015.00 *          at runtime.
0016.00 *          Uses its own custom CPF message: NET9898 which

F3=Exit  F4=Prompt  F5=Refresh  F9=Retrieve  F10=Cursor  F11=Toggle
F16=Repeat find  F17=Repeat change  F24=More keys
(C) COPYRIGHT IBM CORP. 1981, 2013.
M0 + A      MW      05/001
```

```
Create Bound RPG Program (CRTBNDRPG)

Type choices, press Enter.

Program . . . . . > MONOSTDOUT      Name, *CTLSPEC
Library . . . . . > GITTEST123      Name, *CURLIB
Source file . . . . . > QRPGLESRC      Name, QRPGLESRC
Library . . . . . > GITTEST123      Name, *LIBL, *CURLIB
Source member . . . . . > MONOSTDOUT      Name, *PGM
Source stream file . . . . .

Generation severity level . . . 10      0-20
Text 'description' . . . . . *SRCMBRTXT

Default activation group . . . *YES      *YES, *NO

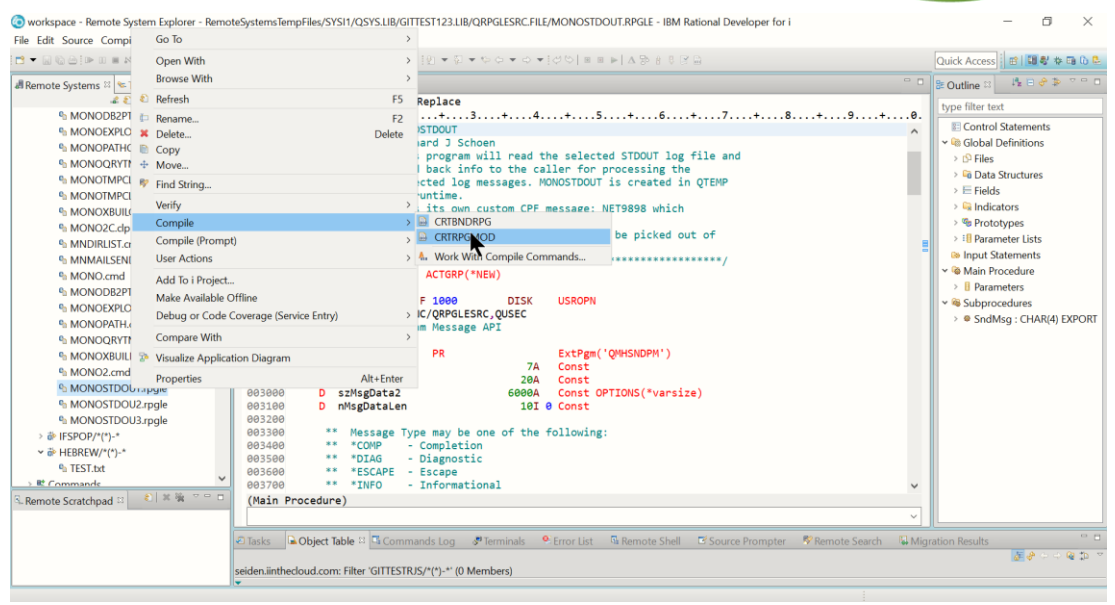
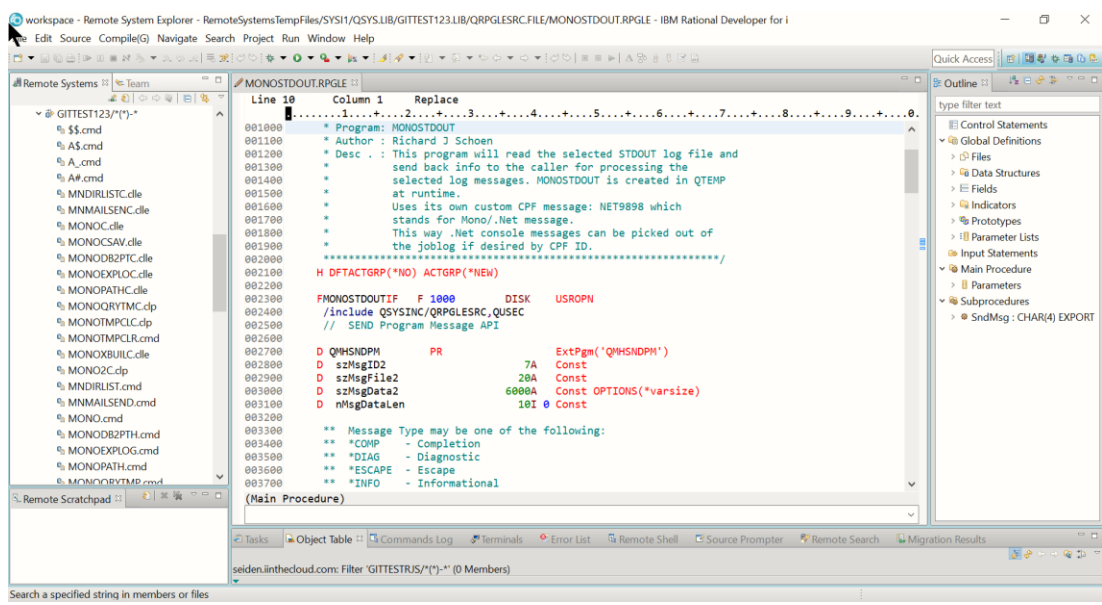
F3=Exit  F4=Prompt  F5=Refresh  F10=Additional parameters  F12=Cancel
F13=How to use this display  F24=More keys

Bottom
M0 + A      MW      05/037
```


Classic RDI IBM i App Development

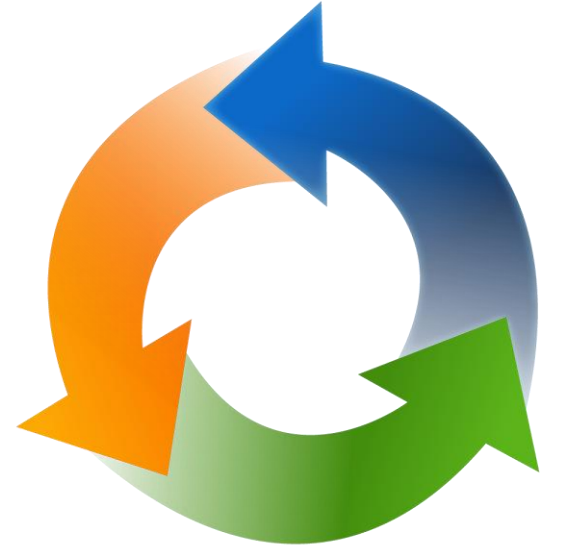


IBM i Source Editing Process



Classic IBM i Development with Git

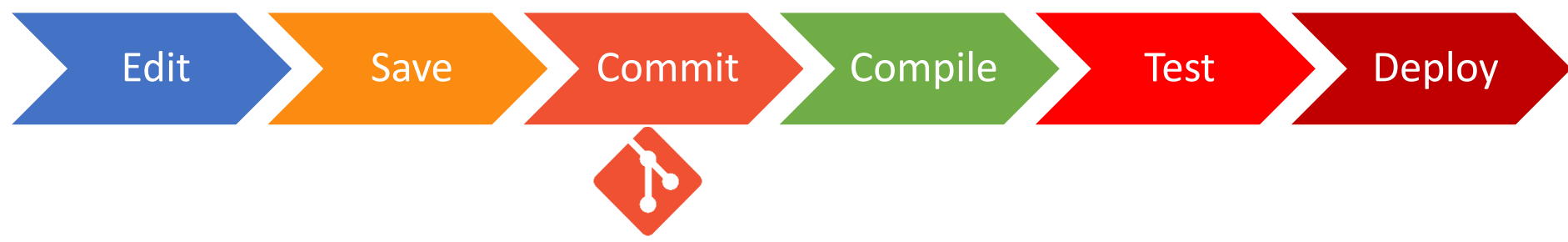
- Edit Source with SEU or RDI
- **Edit**, **Save**, **Commit**
- Compile
- Test
- Repeat...
- Deploy to Production
- Source Library Backups
- **Auto-Commit Changes Hourly, Daily, Weekly, Monthly**



Improved Classic SEU/PDM IBM i Development with iForGit



IBM i Source Editing Process



```
Columns . . . : 6 76      Edit      GITTEST123/QRPGLESRC
SEU==>      MONOSTDOUT
FMT *      *. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+...
***** Beginning of data *****
0001.00 *-----
0002.00 * Source information
0003.00 * @@LIBRARY: GITTEST123
0004.00 * @@FILE: QRPGLESRC
0005.00 * @@MEMBER: MONOSTDOUT
0006.00 * @@TYPE: RPGLE
0007.00 * @@TEXT: Read and parse stdout log....xxxz
0008.00 *-----
0009.00 /*****
0010.00 * Program: MONOSTDOUT
0011.00 * Author : Richard J Schoen
0012.00 * Desc . : This program will read the selected STDOUT log file and
0013.00 *          send back info to the caller for processing the
0014.00 *          selected log messages. MONOSTDOUT is created in QTEMP
0015.00 *          at runtime.
0016.00 *          Uses its own custom CPF message: NET9898 which

F3=Exit  F4=Prompt  F5=Refresh  F9=Retrieve  F10=Cursor  F11=Toggle
F16=Repeat find  F17=Repeat change  F24=More keys
(C) COPYRIGHT IBM CORP. 1981, 2013.
M0 + A      MW      05/001
```

```
Create Bound RPG Program (CRTBNDRPG)

Type choices, press Enter.

Program . . . . . > MONOSTDOUT      Name, *CTLSPEC
Library . . . . . > GITTEST123      Name, *CURLIB
Source file . . . . . > QRPGLESRC      Name, QRPGLESRC
Library . . . . . > GITTEST123      Name, *LIBL, *CURLIB
Source member . . . . . > MONOSTDOUT      Name, *PGM
Source stream file . . . . .

Generation severity level . . . 10      0-20
Text 'description' . . . . . *SRCMBRTXT

Default activation group . . . *YES      *YES, *NO

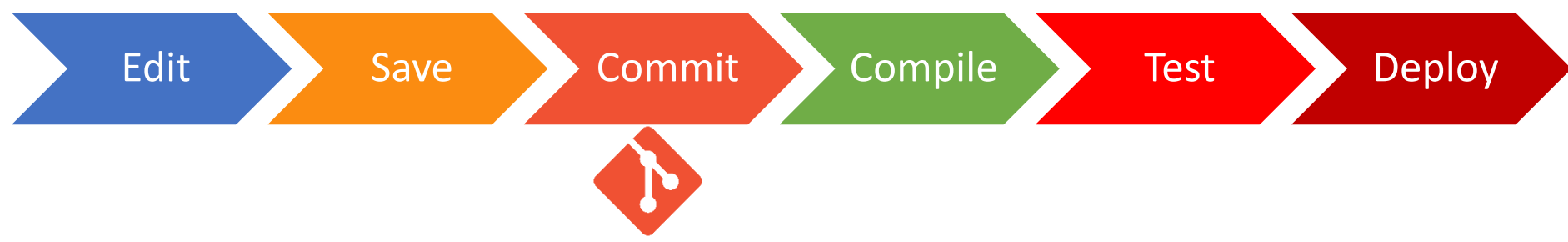
F3=Exit  F4=Prompt  F5=Refresh  F10=Additional parameters  F12=Cancel
F13=How to use this display  F24=More keys

Bottom
M0 + A      MW      05/037
```

Improved Classic SEU/PDM IBM i Development with iForGit



IBM i Source Editing Process



```
Work with Members Using PDM                                SYS1

File . . . . . QRPGLSRC
Library . . . . . GITTEST123      Position to . . . . .

Type options, press Enter.
 2=Edit      3=Copy  4=Delete 5=Display      6=Print      7=Rename
 8=Display description 9=Save 13=Change text 14=Compile 15=Create module...

Opt  Member      Type      Text
GE   MONOSTDOUT  RPGL      Read and parse stdout log....xxz
    MONOSTDOUT2  RPGL      Read and parse stdout log....
    MONOSTDOUT3  RPGL      Read and parse stdout log....

Parameters or command
===>
F3=Exit      F4=Prompt      F5=Refresh      F6=Create
F9=Retrieve   F10=Command entry F23=More options F24=More keys

MB + A MW 11/019
```

```
Export Member to Git Repo Dir (SRCTOGIT)

Type choices, press Enter.

Source file . . . . . QRPGLSRC      Name
Library . . . . . GITTEST123      Name
Source member . . . . . MONOSTDOUT  Character value
Destination git repo IFS dir . . . *LIBREPODTAARA

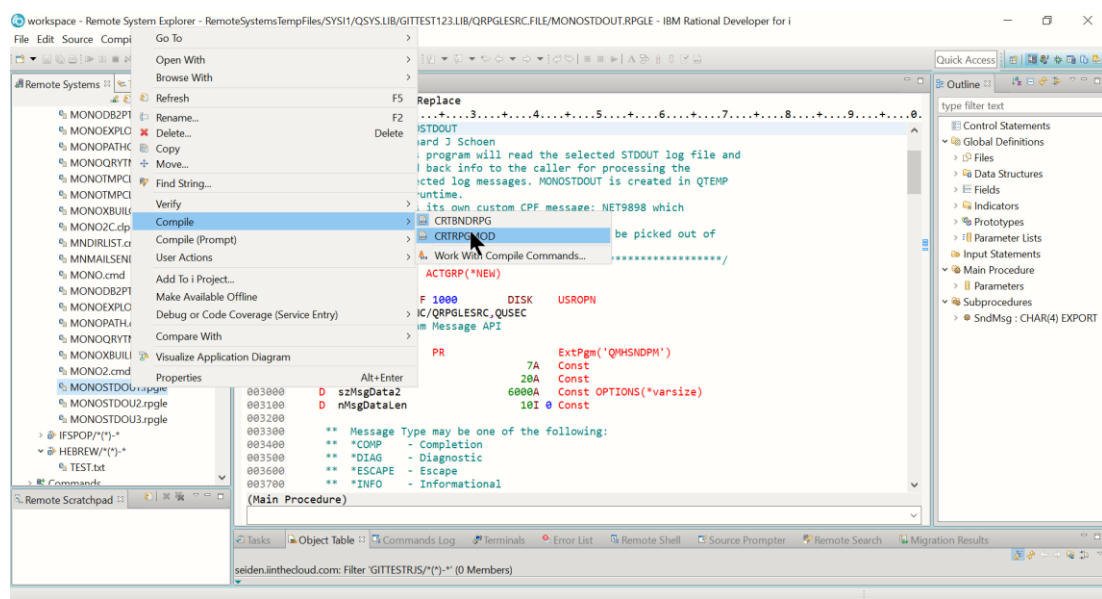
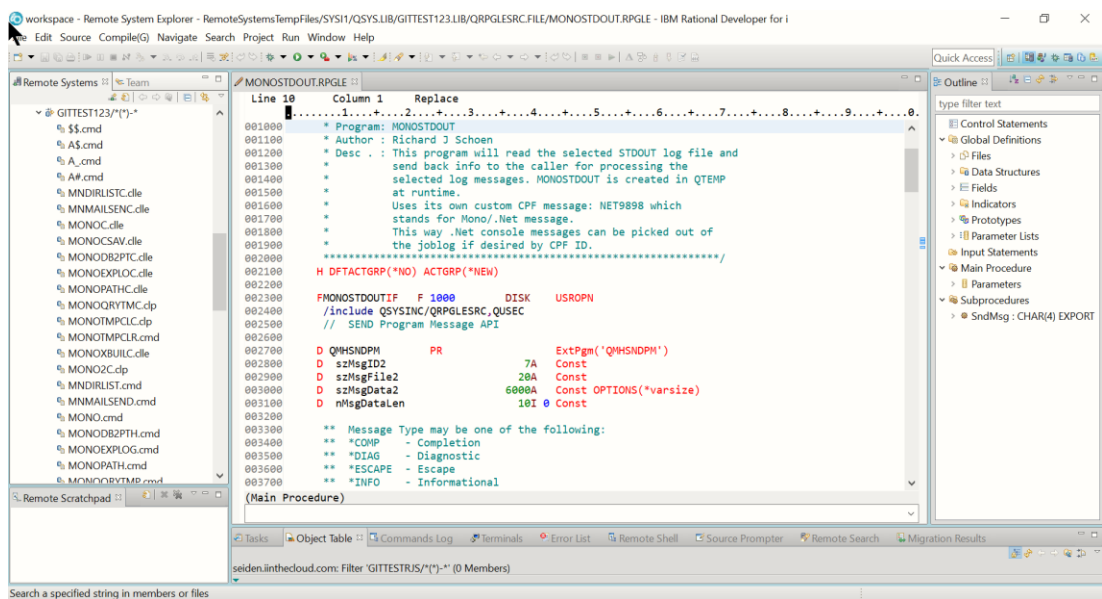
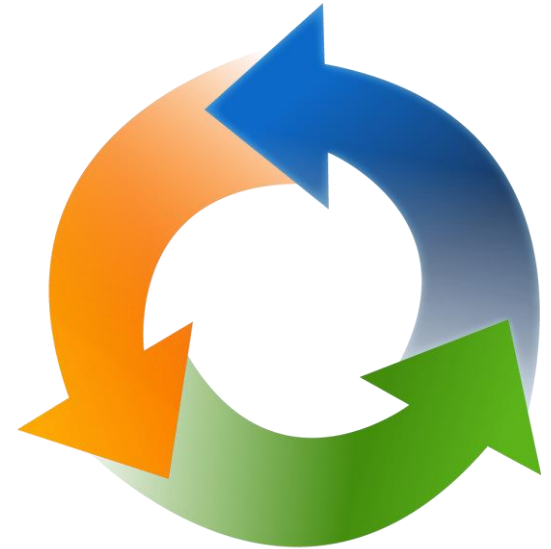
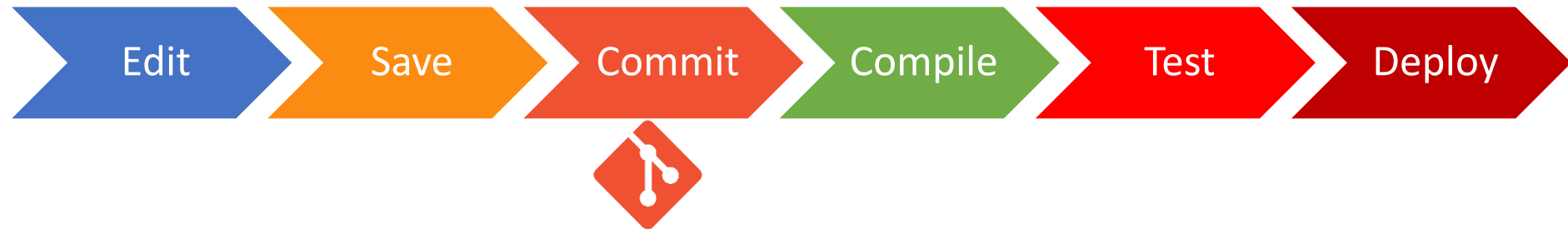
Include source info header . . . > *YES      *NO, *YES
Include source dates and seq . . > *NO      *NO, *YES
Replace existing IFS file mbr . . > *YES      *NO, *YES
Display IFS file after export . . > *NONE     *NONE, *WRKLNK, *DSP, *EDIT
Check if IFS dir is git repo . . > *YES      *NO, *YES
Create IFS git repo dirs . . . . > *YES      *NO, *YES
Init git repo after dir create . . > *YES      *NO, *YES
Commit changes after export . . . > *COMMIT    *COMMIT, *COMMITSYNC...

More...
F3=Exit      F4=Prompt      F5=Refresh      F12=Cancel      F13=How to use this display
F24=More keys

MB + A MW 05/037
```

Improved Classic RDI IBM i Development with iForGit

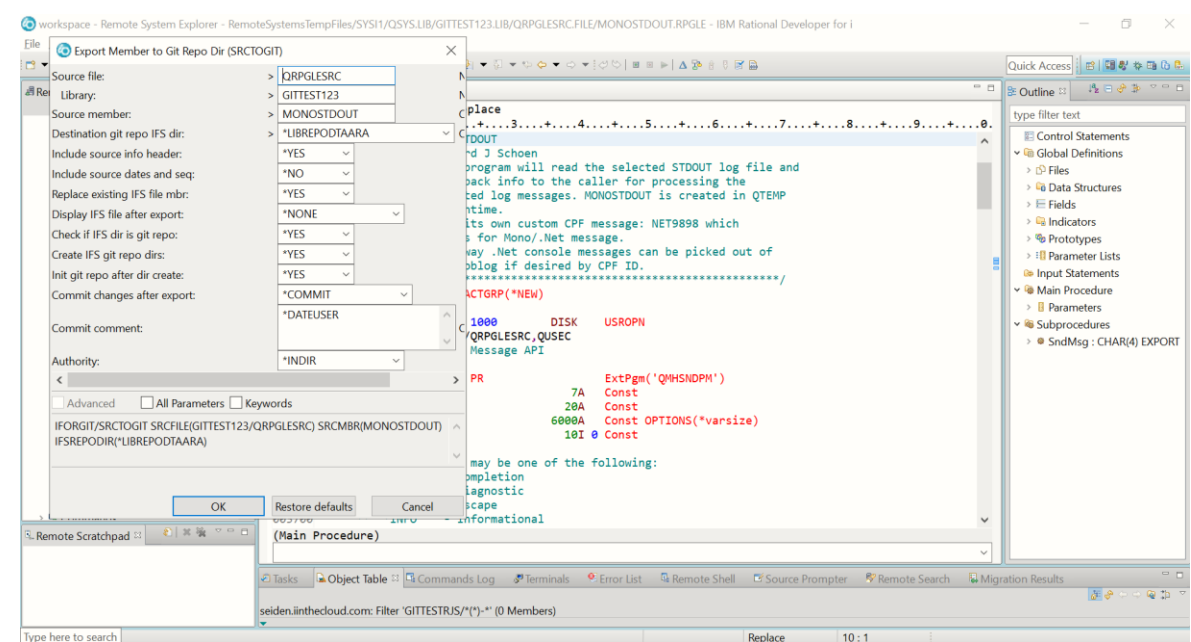
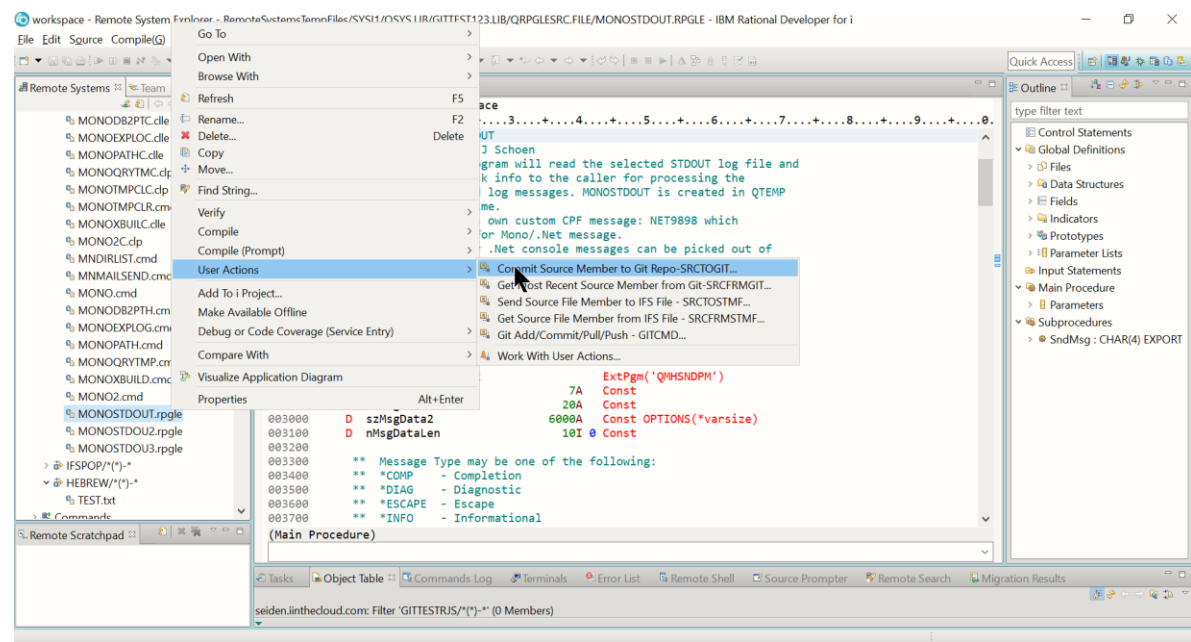
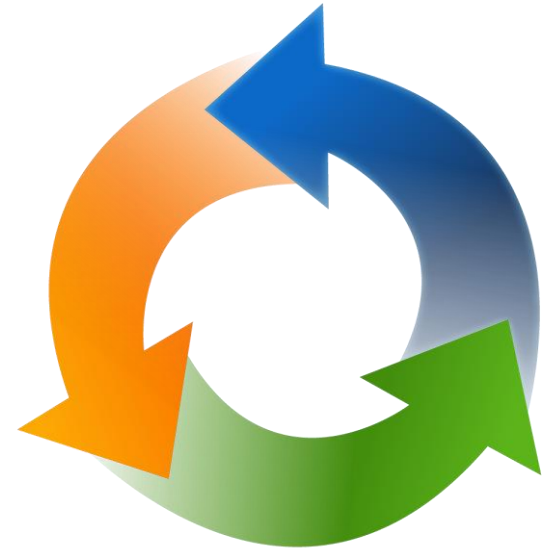
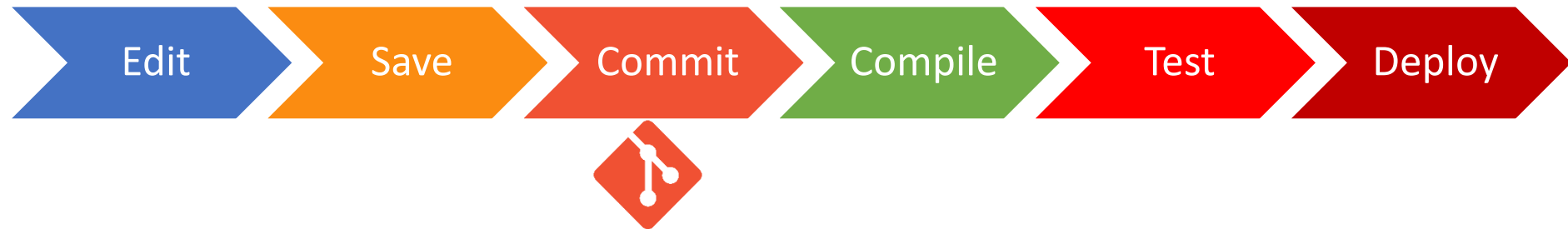
IBM i Source Editing Process with Save to Git



Improved Classic RDI IBM i Development with iForGit



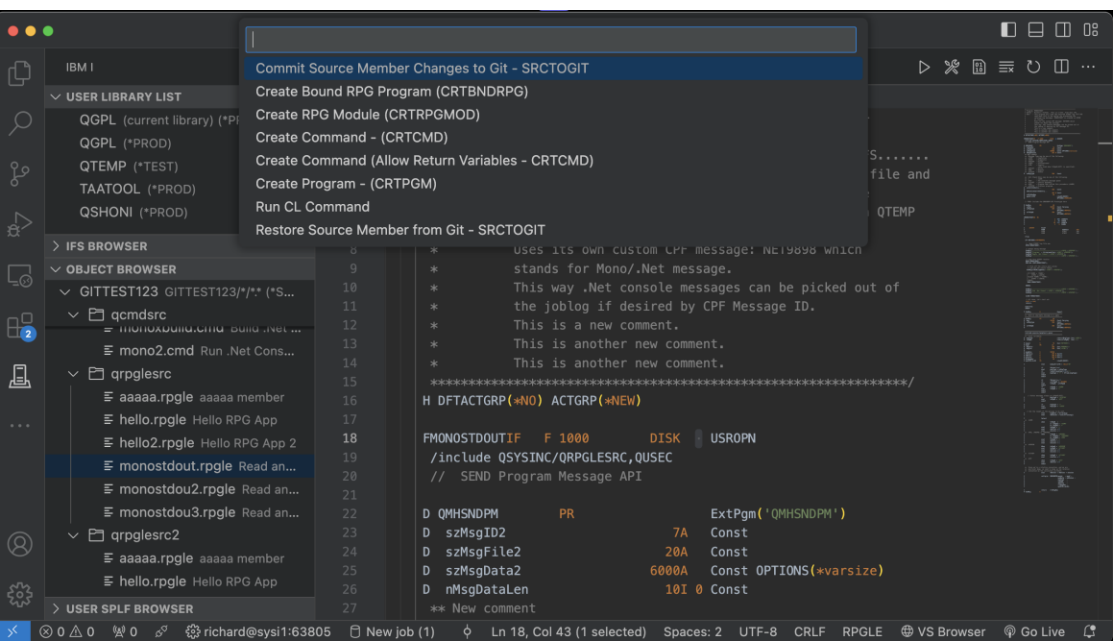
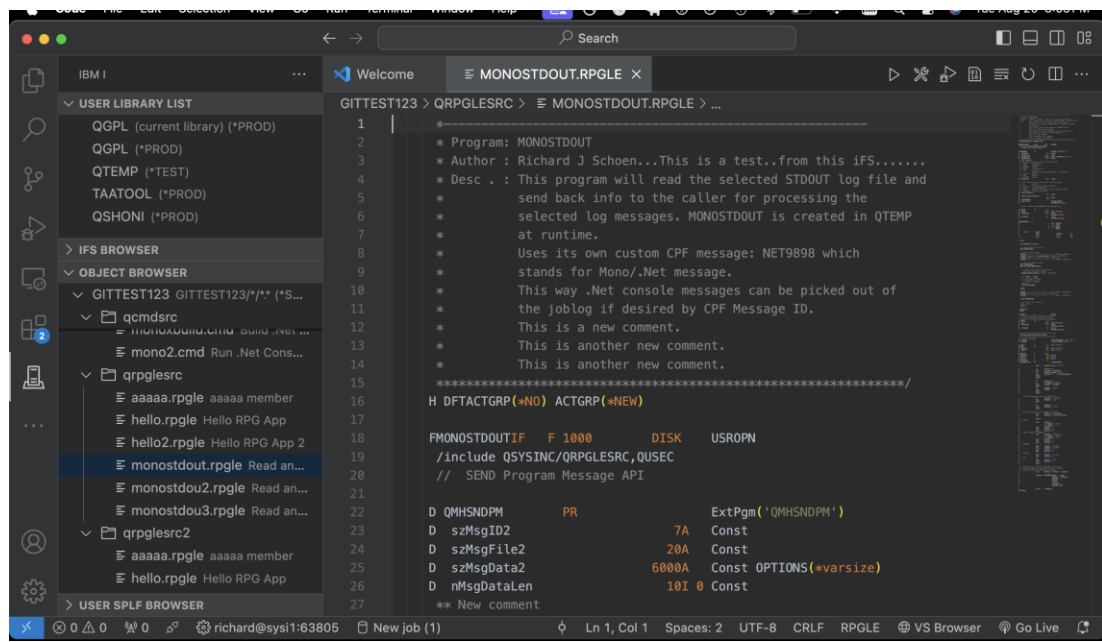
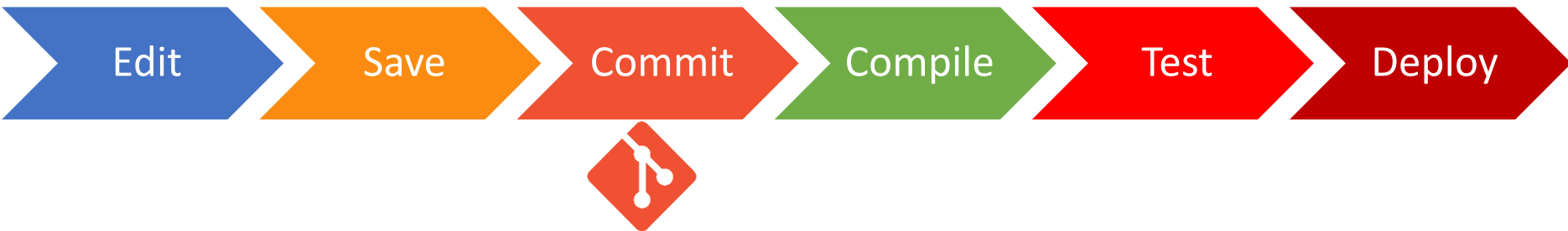
IBM i Source Editing Process with Save to Git



Modern VS Code IBM i Development with iForGit



IBM i Source Editing Process with Save to Git



Other Links

- iForGit Docs

<http://www.mobigogo.net/files/docs/iforgit>

- Git Docs

<https://git-scm.com/docs>

What you learned

- How to start managing IBM i classic source with Git and iForGit
<http://www.mobigogo.net/files/docs/iforgit>
- Install iForGit and Eliminate Awkward Source File Management
- Get started !!
- Questions: richard@richardschoen.net