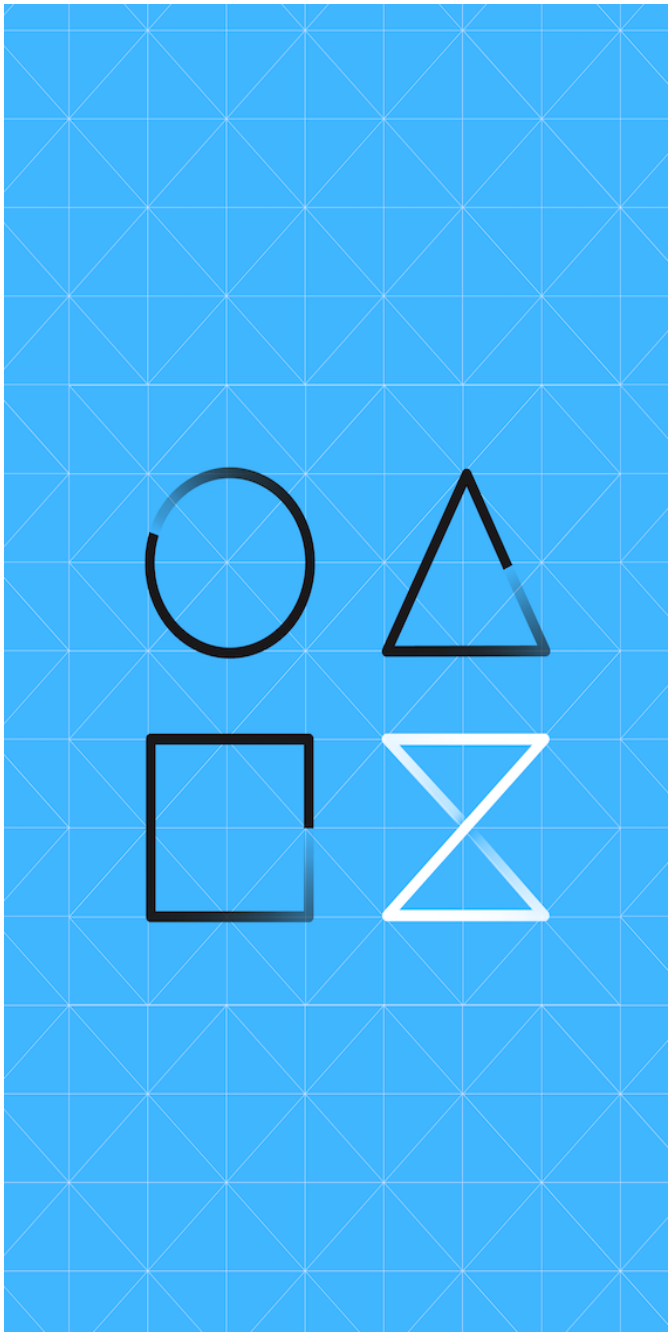


Fundamentals

250117-1355

JCL1

Orchestrating the Enterprise

Tradução automatizada usando o IBM Globalization Service

250117-1355

JCL1 - MAKING THINGS HAPPEN

O Desafio

Você já viu alguns JCL, mas não nos aprofundamos muito.

Nesses desafios, você aprenderá um pouco mais sobre a finalidade do JCL, por que ele é importante em um ambiente Z e como você pode aprimorar ainda mais essas habilidades.

A JCL é uma parte essencial para fazer com que as coisas aconteçam no z/OS, e se familiarizar com os conceitos e a sintaxe permitirá que você supere muitos desafios que poderá enfrentar ao explorar.

Antes De Começar

Você deve ter concluído o desafio FILES1, sobre conjuntos de dados e membros. Se você já entendeu esses conceitos, está pronto para continuar com as etapas deste desafio.

Investimento

Etapas	Duração
17	60 minutos

1 LOAD IT UP

```
zosmf > ZXP.PUBLIC.JCL > ≡ JCLSETUP.jcl > ...
  1   //JCLSETUP JOB ,MSGLEVEL=(0,0)
 19   //*
 20   //* IF YOU REALLY *REALLY* WANT TO START AGAIN, SUBMIT JCLRESET
 21   //*
 22   //*--------------------------------------------------------------
 23   //      EXEC PGM=IEFBR14
 24   //LOAD    DD DSN=&SYSUID..LOAD,DISP=(,CATLG),DATACLAS=SLOAD,
 25   //        SPACE=(TRK,(2,2,2))
 26   //JCL     DD DSN=&SYSUID..JCL,DISP=(,CATLG),DATACLAS=SPDS,
 27   //        SPACE=(TRK,(2,2,2))
 28   //SOURCE DD DSN=&SYSUID..SOURCE,DISP=(,CATLG),DATACLAS=SPDS,
 29   //        SPACE=(TRK,(1,2,2))
 30   //OUTPUT DD DSN=&SYSUID..OUTPUT,DISP=(,CATLG),DATACLAS=SPDS,
 31   //        SPACE=(TRK,(1,2,2))
```

JCL1250117-1355

Procure em **ZXP.PUBLIC.JCL** por um membro chamado **JCLSETUP**.

Esse é um trabalho bastante simples que alocará alguns novos conjuntos de dados necessários para esse e outros desafios.

(A linha nº 26 da captura de tela cria seu próprio conjunto de dados JCL)

Você pode ver na linha nº 24, que menciona &SYSUID..LOAD

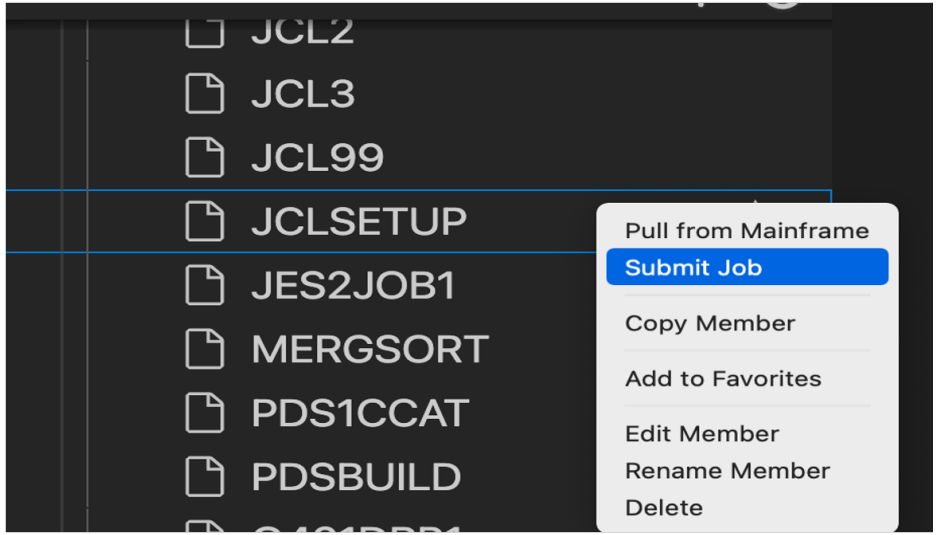
O E comercial (&) com SYSUID depois é conhecido como "Symbolic" e, quando o sistema o vir, ele substituirá automaticamente **&SYSUID.** pelo seu Z-userid.

Observe o "." no final - isso marca o fim do nome simbólico.

Você não precisa fazer nenhuma alteração nesse trabalho para que ele funcione.

Isso significa que todos podem usar o mesmo trabalho e ele substituirá automaticamente &SYSUID. pelo *seu* z-userid. Que conveniente!

2 SUBMIT IT



Clique com o botão direito do mouse nesse trabalho e selecione Submit Job.

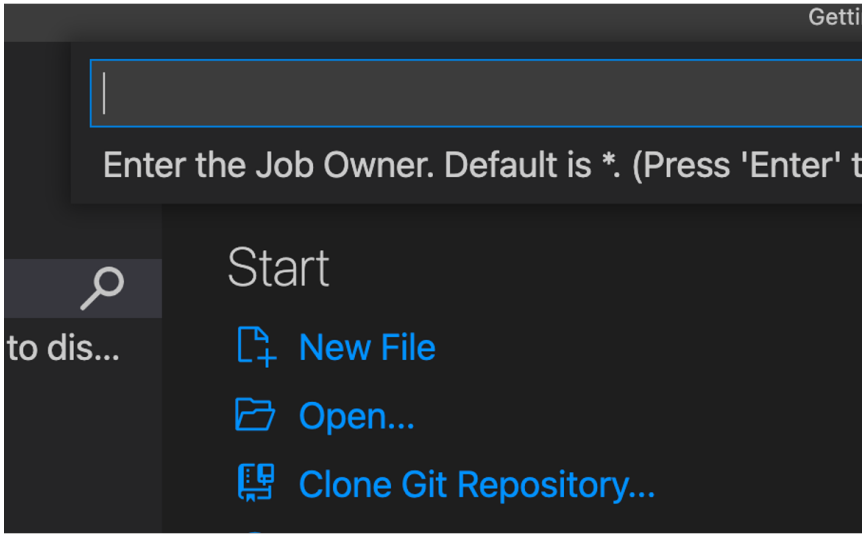
Uma observação sobre a palavra "Job": O JCL é usado para descrever ao sistema exatamente o que você deseja que ele faça.

A tarefa que entregamos ao sistema é conhecida como "Job", e o componente do z/OS que aceita, processa e gerencia a saída desses jobs é conhecido como **Job Entry Subsystem (JES)**.

Portanto, para este desafio, você enviou um trabalho para JES para que ele processasse a tarefa que você acabou de ver.

Observação : esse trabalho tem o objetivo de criar conjuntos de dados para você e pressupõe que você ainda não tenha esses conjuntos de dados; *se você executá-lo mais de uma vez, provavelmente verá erros sobre nomes de conjuntos de dados* ***DUPLICADOS***.

3 FILTER AND FIND



Você já deve ter um perfil em **JOBS** no lado esquerdo do VSCode.

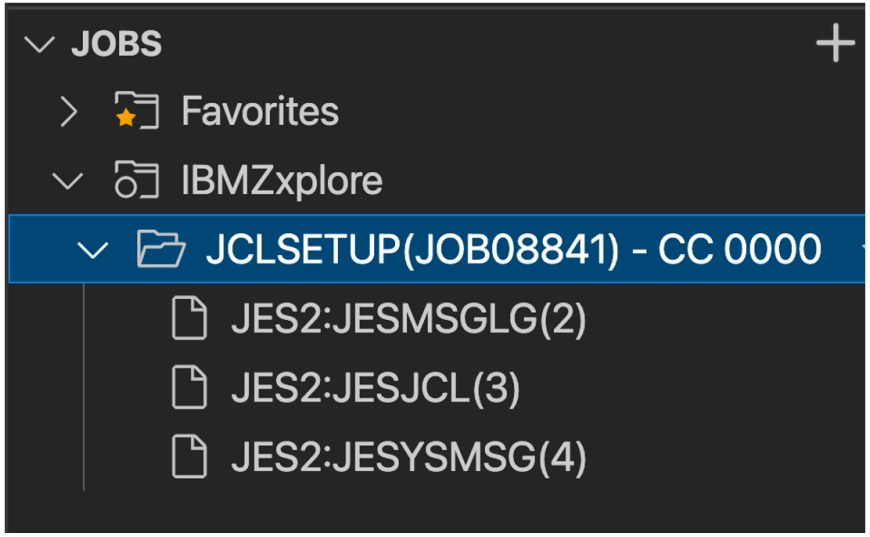
Clique na lupa ( ) à direita dela.

- Digite seu ID de usuário para o Job Owner
- Digite um asterisco (*) para o prefixo do trabalho
- Pressione Enter (em branco, sem dados) para a Job Id Search.

Você pode refazer esse filtro clicando novamente na lupa e selecionando Owner/Prefix ou Job Id. Você deve conseguir encontrar o trabalho que acabou de enviar aqui. Procure por **JCLSETUP**.

A próxima etapa abordará isso com mais detalhes.

4 YOU GOT A ZERO. PERFECT!

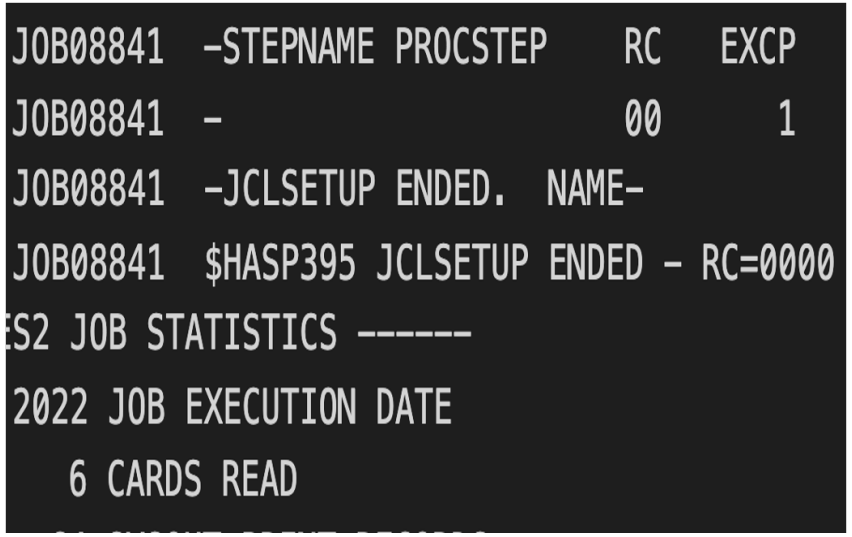


Abra o triângulo "twistie" ao lado do trabalho JCLSETUP que você acabou de enviar. Provavelmente também haverá outros empregos, mas você está procurando especificamente por **JCLSETUP**. Se você a enviou mais de uma vez, encontre a que tem o endereço CC 0000 à direita.

Você também verá esse número no **JESMSGLG** quando você abrir o twistie. Um código de condição (CC) de zero significa que tudo foi executado conforme o esperado, sem erros, o que é bom!

Se você obtiver qualquer outro número para o código de conclusão, geralmente há algo que vale a pena investigar - e provavelmente consertar.

5 JUMP RIGHT TO IT



Você deve ter notado que, após enviar JCL em VSCode, uma pequena mensagem aparece no canto inferior direito da janela VSCode.

Em vez de vasculhar a saída do site JOBS, você geralmente pode clicar na mensagem e ela o levará diretamente para a saída.

Um trabalho começará em **ACTIVE** enquanto estiver sendo executado.

Você pode atualizar o status de um trabalho fechando e reabrindo a chave de fenda à esquerda do nome do trabalho.

POR QUE O JES E O JCL SÃO IMPORTANTES? POR QUE NÃO POSSO SIMPLESMENTE EXECUTAR PROGRAMAS?

Quando você envia JCL, ele vai para o Job Entry Subsystem (abreviado para JES).

JES examina o site JCL que você enviou e reúne todos os recursos necessários para realizar a tarefa. Em um sistema com carga pesada, pode ser necessário priorizar alguns trabalhos em níveis mais baixos ou mais altos do que outros para que o trabalho importante seja feito mais rapidamente.

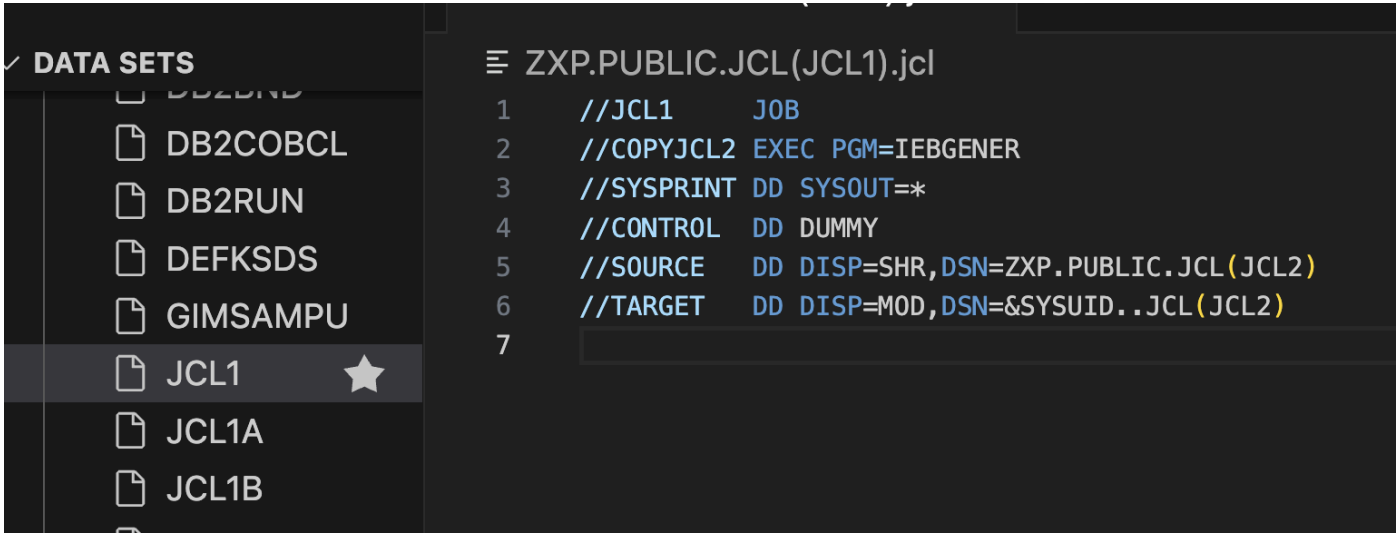
Pense em JCL como o pedido que o garçom faz e em JES como a equipe da cozinha que analisa o pedido e decide como será feito.

O L em JCL significa Language (Linguagem), mas na verdade não se trata de uma linguagem de programação, mas sim de uma maneira de descrevermos efetivamente as tarefas para o sistema.

Tudo o mais que aparece na saída do trabalho (o "registro do trabalho") são informações sobre como o trabalho foi executado. Como você pode ver, há muitas informações aqui.

6 MY FIRST COPY JOB

Agora você tem mais alguns conjuntos de dados com os quais trabalhar e, como este é um desafio do JCL, você precisa obter alguns de seus próprios JCL para trabalhar.



JCL1I250117-1355

Copie o **JCL1** membro de **ZXP.PUBLIC.JCL** para seu próprio **JCL** conjunto de dados e, em seguida, abra sua cópia. É *necessário* ter isso em seu próprio conjunto de dados JCL, pois você o modificará nas próximas etapas.

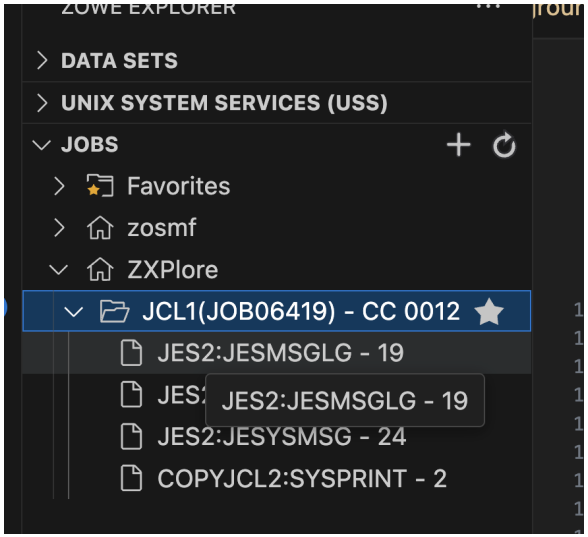
Talvez seja necessário fechar e reabrir o triângulo **DATA SETS** para atualizar a visualização, de modo que o site JCL1 apareça.

Esse trabalho é fornecido para que você possa copiar o membro *JCL2* de ZXP.PUBLIC.JCL para seu conjunto de dados JCL, mas usando o **IEBGENER** em vez de VSCode.

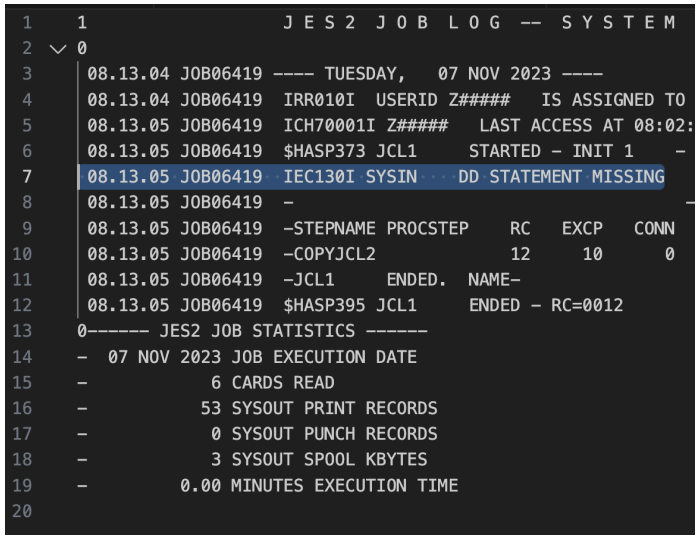
Envie JCL1 da mesma forma que enviou o JCLSETUP e examine o registro de trabalho.

7 FIRST ERROR

Ao consultar a seção JOBS e ver o trabalho JCL1, você verá imediatamente que o código de conclusão é **0012** ; presuma que isso significa que algo deu errado.



Abra a seção **JES2:JESMSGLG** seção do trabalho e verifique se há uma indicação do que causou a falha.



Nesse caso, você pode ver que o problema se deve à falta de uma instrução DD para **SYSIN**

Em JCL, as declarações que definem de onde os dados vêm ou para onde vão são conhecidas como declarações **de definição de dados** ou, simplesmente, "declarações DD ".

Nesse trabalho, há apenas duas etapas - ambas executam o programa **IEBGENER** se você pesquisar on-line, encontrará as seguintes informações sobre a configuração do IEBGENER :

https://www.ibm.com/docs/en/zos/3.1.0?topic=c-job-control-statements-4

Arquivo/DD	Finalidade
SINALIZAÇÃO	Define um conjunto de dados sequenciais para mensagens. O conjunto de dados pode ser gravado em um dispositivo de saída do sistema, em um volume de fita ou em um volume DASD
SYSUT1	Define o conjunto de dados de entrada. A "fonte". Ele pode fazer referência a um conjunto de dados sequenciais existente, a um membro de um conjunto de dados particionado ou PDSE ou a um arquivo UNIX z/OS. Um conjunto de dados sequenciais pode ser de formato básico, formato grande, formato estendido, formato

Arquivo/DD	Finalidade
	compactado, entrada em spool, fita, leitor de cartão, terminal TSO ou DUMMY. Observe que **a fonte deve existir** - se não existir, o IEBGENER terminará com um erro **013**.
SYSUT2	Define o conjunto de dados de saída. O "alvo". Ele pode definir um conjunto de dados sequenciais, um membro de um conjunto de dados particionado ou PDSE, um conjunto de dados particionado ou PDSE ou um arquivo UNIX z/OS. Um conjunto de dados sequenciais pode ser de formato básico, formato grande, formato estendido, formato compactado, saída em spool, fita, perfurador de cartão, impressora, terminal TSO ou DUMMY
SYSIN	Define o conjunto de dados de controle ou especifica DUMMY quando a saída é sequencial e nenhuma edição é especificada. O conjunto de dados de controle normalmente reside no fluxo de entrada; no entanto, ele pode ser definido como um membro de um conjunto de dados particionado ou PDSE.

JCL1i250117-1355

Esperamos que você possa ver que os arquivos atribuídos para IEBGENER no trabalho JCL1, usando as instruções DD, não correspondem aos arquivos **exigidos** pelo programa.

Faça as alterações apropriadas nas instruções DD em seu membro JCL1**(lembre-se de salvar!**) e enviar novamente.

Se você tiver feito as alterações corretas, o trabalho deverá ser concluído com CC=0000.

Caso contrário, verifique novamente o registro de trabalho em busca de mensagens que indiquem a causa dos novos erros; faça os ajustes necessários e tente novamente.

Com esse tipo de programa, é fácil identificar e corrigir o que deu errado, como declarações DD incorretas ou ausentes - está no livro!

Após a conclusão bem-sucedida do trabalho JCL1, você deverá ter um novo **JCL2** em seu conjunto de dados JCL.

(Você também terá um membro **JCL3** que é criado pela segunda etapa do trabalho JCL1)

8 KICKING OFF SOME COBOL

```
1   //JCL2    JOB 1
2   //*************************************************/
3   //COBRUN  EXEC IGYWCL
4   //COBOL.SYSIN  DD DSN=ZXP.PUBLIC.SOURCE(CBL0001),DISP=SHR
5   //LKED.SYSLMOD DD DSN=&SYSUID..LOAD(CBL0001),DISP=SHR
6   //*************************************************/
7   // IF RC = 0 THEN
8   //*************************************************/
9   //RUN     EXEC PGM=CBL0001
10  //STEPLIB   DD DSN=&SYSUID..LOAD,DISP=SHR
11  //FNAMES    DD DSN=ZXP.PUBLIC.INPUT(FNAMES),DISP=SHR
12  //LNAMES    DD DSN=ZXP.PUBLIC.INPUT(LNAMES),DISP=SHR
13  //COMBINE   DD DSN=&SYSUID..OUTPUT(NAMES),DISP=SHR
14  //SYSOUT    DD SYSOUT=*,OUTLIM=15000
15  //CEEDUMP   DD DUMMY
16  //SYSUDUMP  DD DUMMY
```

JCL1I250117-1355

Edite sua cópia pessoal da **JCL2** definição do trabalho.

Talvez seja necessário fechar e reabrir o triângulo **DATA SETS** para atualizar a visualização, de modo que o site JCL2 apareça.

Esse JCL é usado para compilar e executar alguns códigos do site COBOL. Após a compilação, ele colocará o programa resultante em seu **LOAD** conjunto de dados.

Observação : os programas no conjunto de dados LOAD são binários - você não conseguirá visualizar nada aqui com o VSCode.

Procure a linha que começa com **//COBRUN** - esse é o início de uma "etapa" do trabalho que executará o compilador COBOL.

Na próxima linha, você pode ver o conjunto de dados de entrada (a fonte) na linha 18 (**//COBOL.SYSIN**) e onde ele colocará a saída na linha seguinte (**//LKED.SYSLMOD**).

As linhas que seguem o início da etapa **//RUN** têm um formato semelhante:

```
//ddname  DD DSN=dataset,DISP=access
```

- "ddname" também é conhecido como nome de arquivo - o nome usado por programas para acessar dados em conjuntos de dados
- "dataset" é o local real dos dados - isso pode mudar, mas o programa não precisa estar ciente
- "acesso" (ou "disposição") indica como o programa pode usar o conjunto de dados

Lendo mais adiante, se a etapa de trabalho terminar com um código de conclusão 0 (porque não houve problemas) da etapa de compilação, ela executará o programa **CBL0001** programa.

Tudo está fazendo sentido até agora? Você usará o JCL para compilar o código-fonte do COBOL e, em seguida, executar o programa resultante.

O QUE SIGNIFICA COMPILAR? O QUE É COBOL?

COBOL é uma linguagem de programação usada em muitas instituições financeiras, de saúde e governamentais. Seu alto grau de precisão matemática e métodos de codificação simples fazem dele uma opção natural quando os programas precisam ser rápidos, precisos e fáceis de entender.

O código que é escrito por seres humanos precisa ser transformado em código de máquina para ser executado como um programa. A compilação é uma etapa que realiza essa transformação. Nosso site JCL tem duas etapas principais: compilar o código-fonte em código de máquina e, em seguida, executar o programa.

Esse programa específico também requer dois arquivos de entrada e grava em um arquivo de saída, portanto, também especificaremos esses arquivos (conjuntos de dados) no site JCL.

9 RUN, CODE, RUN

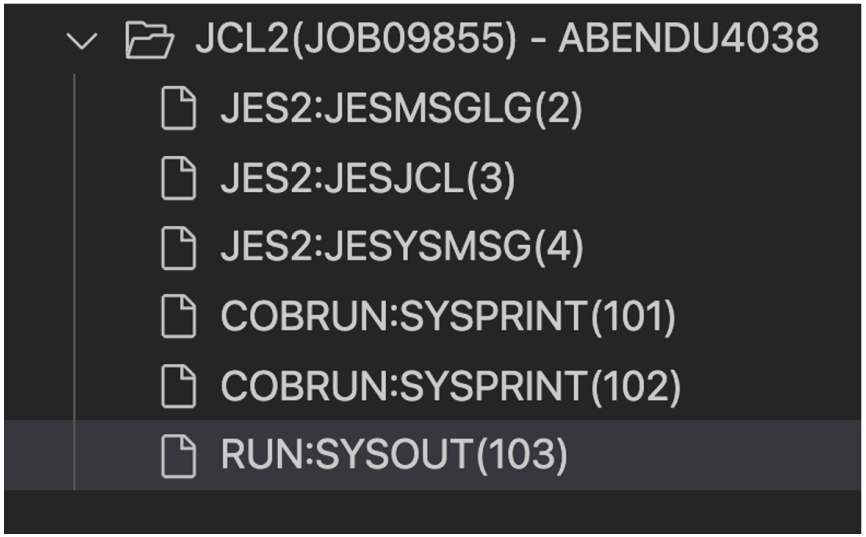
Após a compilação bem-sucedida do código COBOL, o site JES executará o programa (o comando **//RUN EXEC PGM=CBL0001** e informará onde encontrar os conjuntos de dados de entrada, bem como onde a saída será armazenada em seu conjunto de dados OUTPUT -

```
//COMBINE    DD DSN=&SYSUID..OUTPUT(NAMES),DISP=SHR
```

O nome do comando DD é o que vem logo após as barras duplas, portanto, " FNAMES " e " LNAMES ", por exemplo.

Todas as linhas que começam com **//*** são comentários e são ignoradas pelo site JES. As linhas comentadas são úteis para fornecer informações informativas ou manter linhas de código que poderão ser usadas posteriormente, mas que não são necessárias no momento.

10 THEY CAN'T ALL BE ZEROES



Enviar **JCL2** do seu conjunto de dados JCL e, em seguida, examine a saída, usando o que você aprendeu nas etapas anteriores deste desafio.

OBSERVAÇÃO: Você *receberá* um **ABEND** (abreviação de Abnormal End), o que significa que algo ainda não está certo.

Mas não se preocupe - com suas novas habilidades, você chegará ao fundo da questão!

Nas etapas anteriores do trabalho JCL1, você poderia usar a documentação do **IEBGENER** para determinar quais instruções DD eram *necessárias* para que o programa funcionasse; nesse caso, não há documentação, apenas o próprio código COBOL.

Na próxima etapa, você examinará o código COBOL e verá como o código real corresponde ao código JCL que está sendo usado para compilá-lo e executá-lo.

E, novamente, não se preocupe! Você não precisa se tornar um especialista em COBOL para resolver isso - lembre-se de que este é um **JCL** desafio, não um desafio COBOL.

Para que os programas do z/OS trabalhem com conjuntos de dados em um trabalho, eles precisam de quatro coisas:

1. a definição interna do arquivo que representa a estrutura de dados e o conteúdo que o programa criará, lerá, modificará ou excluirá - isso é definido dentro do programa
2. o conjunto de dados atual que uma execução específica do programa pode usar - cada vez que o programa é executado, ele pode usar conjuntos de dados diferentes, desde que tenham o formato correto esperado pelo programa; esse é o nome usado no parâmetro **DSN=** de uma instrução DD
3. uma escolha correta de disposição para o conjunto de dados para indicar se ele deve ser criado, excluído ou repassado - esse é o parâmetro ** `DISP=* *` de uma declaração DD
4. uma instrução DD que vincula o arquivo interno do programa ao conjunto de dados específico - isso deve corresponder ao nome da definição de arquivo do programa e especificar o conjunto de dados necessário e a disposição

11 COMPARE THE CODE

```
*-----------------------
 IDENTIFICATION DIVISION.
*-----------------------
 PROGRAM-ID.    NAMES
 AUTHOR.        Otto B. Named
*--------------------
 ENVIRONMENT DIVISION.
*--------------------
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT FIRST-NAME ASSIGN TO FNAMES.
     SELECT LAST-NAME  ASSIGN TO LNAMES.
     SELECT FIRST-LAST ASSIGN TO COMBINED.
```

A instrução JCL que começa com **//COBOL.SYSIN** aponta para o conjunto de dados do código-fonte COBOL que será compilado, portanto, comece por aí. Abra o código-fonte do programa em VSCode e comece examinando a área **FILE-CONTROL** área.

É aqui que você obtém os nomes usados pelo programa que precisam ser combinados no JCL. Por exemplo, **FIRST-NAME** é uma referência de registro no código COBOL para um arquivo e que é atribuído (ou vinculado) à instrução **FNAMES** DD no JCL.

Abra o código JCL, COBOL e o registro de trabalho - examine a saída e você poderá ver o que

*alteração muito simples de **uma única letra***

precisa ser feito no trabalho **JCL2** trabalho para que tudo se conecte entre o programa COBOL, os conjuntos de dados e o JCL.

Quando você tiver corrigido o problema em JCL2, ele deverá ser executado com um CC=0000 e você encontrará *a saída correta no membro correto* do conjunto de dados OUTPUT.

12 ALL ABOARD

```
//*
//PEEKSKL EXEC PGM=IEBGENER
//SYSPRINT DD DUMMY
//SYSIN    DD DUMMY
//SYSUT1   DD *
Peekskill - 41mi
//SYSUT2   DD DSN=&SYSUID..JCL3OUT,DISP=(MOD,PASS,DELETE),
//            SPACE=(TRK,(1,1)),UNIT=SYSDA,
//            DCB=(DSORG=PS,RECFM=FB,LRECL=80)
//*
//CORTLNDT EXEC PGM=IEBGENER
//SYSPRINT DD DUMMY
//SYSIN    DD DUMMY
//SYSUT1   DD *
Cortlandt - 38mi
```

JCL1I250117-1355

Abra o membro **JCL3** do seu conjunto de dados JCL e dê uma olhada no que está lá dentro. Você verá que essa JCL contém várias etapas, cada uma delas usando o **IEBGENER** para direcionar um ou mais registros para um conjunto de dados sequenciais.

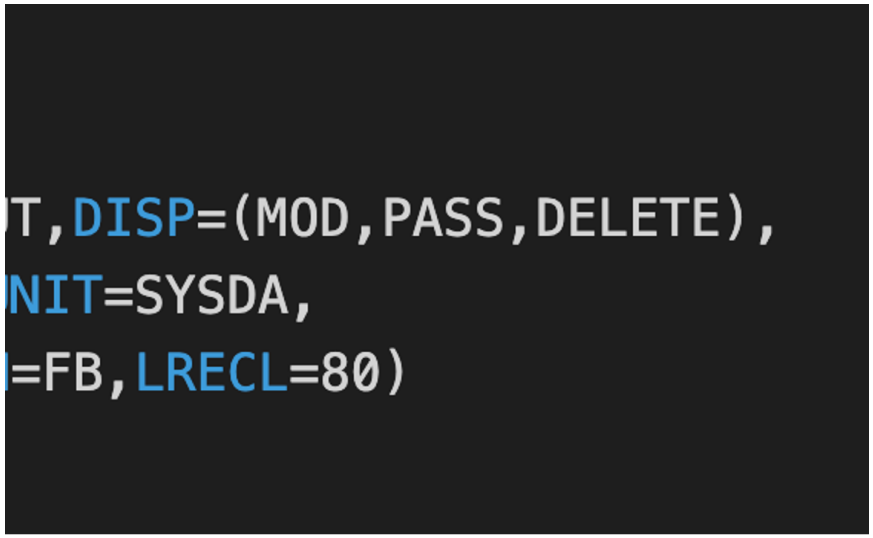
Essa é uma extensão bastante simples do trabalho JCL1 que você trabalhou anteriormente - uma grande diferença aqui é como os dados de "origem" são definidos.

Nesse trabalho, os dados para todas as **SYSUT1** são definidos como "in-stream", ou seja, estão ali mesmo no trabalho, após a instrução DD. Isso é especificado pela opção **DD *** em vez de um parâmetro **DSN=** parâmetro. O programa IEBGENER copiará a entrada de SYSUT1 até que a próxima instrução JCL seja alcançada - começando com **//** ou **/***

Você verá que há uma linha de cabeçalho, algumas linhas com informações da estação para as paradas de trem de pico entre Poughkeepsie, NY e o Grand Central Terminal em Nova York, seguidas de algum texto sobre o horário de funcionamento.

Parece bem simples... qual será a parte mais complicada?

13 A FRIENDLY DISPOSITION



Em cada parte da JCL que você usou até agora, você deve ter encontrado algum tipo de parâmetro **DISP=** (disposição).

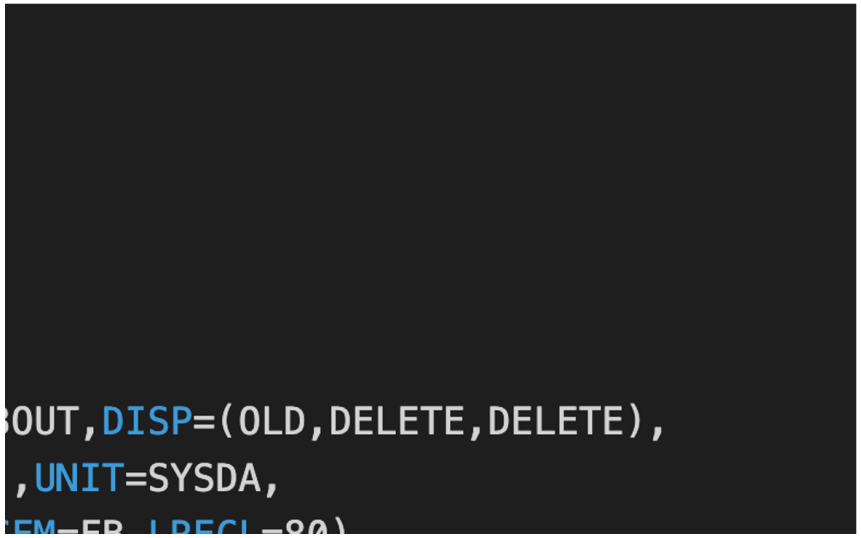
DISP são usados para descrever como o JCL deve usar ou criar um conjunto de dados e o que fazer com ele após a conclusão do trabalho ou da etapa do trabalho.

Um parâmetro padrão do site DISP tem três partes. O primeiro parâmetro é o status, que pode ser qualquer um dos seguintes:

Parâmetro	Significado
NOVO	Criar um novo conjunto de dados
SHR	Reutilizar um conjunto de dados existente e permitir que outras pessoas o utilizem, se desejarem
VELHO	Reutilize um conjunto de dados existente, mas não permita que outros o utilizem enquanto nós o estivermos usando
MOD	

Parâmetro	Significado
	Somente para conjuntos de dados sequenciais. Reutilizar um conjunto de dados existente, mas apenas anexar novos registros à parte inferior dele. Se não houver um conjunto de dados, crie um novo.

14 WHAT'S YOUR STATUS?



O campo 2 do parâmetro DISP descreve o que deve acontecer com o conjunto de dados no caso de uma conclusão normal da etapa de trabalho, e o terceiro campo é o que deve acontecer com o conjunto de dados no caso de uma falha na etapa de trabalho.

Há vários valores que podem ser usados aqui, mas, para este desafio, você só precisa conhecer os seguintes:

Field2	Significado
DELETE	Apague-o completamente do armazenamento
CATLG	Registre o conjunto de dados para que possa usá-lo após o término do trabalho
PASS	Depois que essa etapa for concluída, mantenha-a para que as etapas de trabalho posteriores a ela (no mesmo trabalho) possam usá-la

15 YOU'RE NO DUMMY

```
//JCL3     JOB
//*
//* IEBGENER is a system utility program to copy data
//*  where the default input  filename is SYSUT1
//*  and   the default output filename is SYSUT2
//*
//HEADER EXEC PGM=IEBGENER
//SYSPRINT DD DUMMY
//SYSIN    DD DUMMY
//SYSUT1   DD *
*****************************************
METRO NORTH POUGHKEEPSIE -> NYC M-F SCHEDULE
PEAK HOUR OPERATION
*****************************************
```


Você deve ter notado muitas menções de **DUMMY** nas declarações de DD.

Não se preocupe, esta JCL não está xingando ninguém; é apenas uma maneira de dizer "Esta alocação de arquivo é necessária, mas desta vez não será usada para nada, portanto, não importa".

Como você viu anteriormente, o programa **IEBGENER** executado em cada etapa requer:

- **SYSIN** uma instrução de arquivo de entrada (DD) para comandos de controle
- **SYSPRINT** uma instrução DD de saída para que o programa informe sucesso, falha, progresso
- **SYSUT1** uma instrução DD de "origem" de onde os dados devem ser copiados
- **SYSUT2** um comando DD de "destino" para o qual os dados de SYSUT1 devem ser copiados

Mas, nesse trabalho, a saída do programa não é necessária e não há instruções de controle necessárias, portanto, DUMMY é uma forma de dizer "Não importa, não perca seu tempo configurando isso" - basta executar a ação padrão e copiar o conjunto de dados associado a **SYSUT1** para o conjunto de dados associado a **SYSUT2**.

16 RIGHT ON TIME

Envie sua cópia do trabalho **JCL3** e veja o resultado.

OBSERVAÇÃO: você deve esperar que essa tarefa seja concluída com o código 0000 - isso significa que nada está quebrado - mas não significa que a saída esteja correta!

Há duas edições que você precisa fazer para que esse trabalho seja executado 100% corretamente:

- uma lista completa de **todas as 10 paradas de estações**, de Poughkeepsie ao Grand Central Terminal
- as informações na parte superior e inferior do conjunto de dados.
- Você não deve ter ***paradas repetidas***. Se Beacon ou Cortlandt estiver listado lá duas vezes, algo ainda precisa ser consertado.

NOTA: *Você precisará excluir o* ***JCL3OUT*** *conjunto de dados de saída sempre antes de reenviar* ***JCL3***

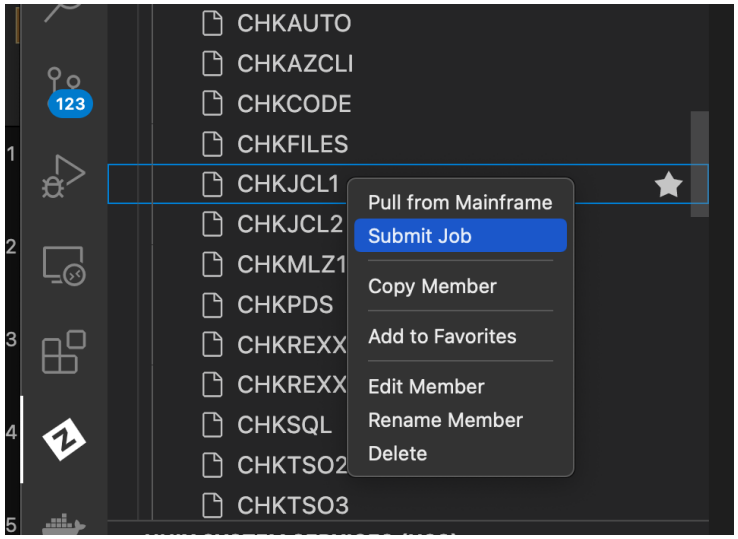
Opcionalmente, verifique novamente o JCL dos trabalhos JCLSETUP e PDSBUILD para ver como você pode excluir automaticamente o JCL3OUT adicionando uma etapa extra no início do JCL3.

Envie JCL3 novamente e verifique se você tem o resultado correto.

Quando concluído, você deverá ter a saída completa em seu **JCL3OUT** conjunto de dados sequenciais, totalizando 23 linhas (registros) - o mesmo que a visualização a seguir:

```
*******************************************
METRO NORTH POUGHKEEPSIE -> NYC M-F SCHEDULE
PEAK HOUR OPERATION
*******************************************
Poughkeepsie - 74mi
New Hamburg - 65mi
Beacon - 59mi
Cold Spring - 52mi
Garrison - 50mi
Peekskill - 41mi
Cortlandt - 38mi
Croton-Harmon - 33mi
Harlem - 125th Street - 4mi
Grand Central Terminal - 0mi
************************************************************
Peak fares are charged during business rush hours on any
weekday train scheduled to arrive in NYC terminals between
6 a.m. and 10 a.m. or depart NYC terminals between 4 p.m.
and 8 p.m. On Metro-North trains, peak fares also apply to
travel on any weekday train that leaves Grand Central Terminal
between 6 a.m. and 9 a.m.
Off-peak fares are charged all other times on weekdays, all
day Saturday and Sunday, and on holidays.
```

17 WHO KNEW?



Agora envie o trabalho **CHKJCL1** de **ZXP.PUBLIC.JCL** para validar a saída correta de JCL2 e JCL3, e espere ver o código de conclusão (CC) de 0000.

Se CHKJCL1 retornar CC=0127, volte e verifique novamente e ajuste seu trabalho para JCL2 e JCL3, e envie esses trabalhos novamente, se necessário.

Verifique novamente se a saída está correta, enviando **CHKJCL1** novamente até obter CC=0000.

MAIS UMA VEZ: *Você precisará se certificar de que o conjunto de dados de saída* ***JCL3OUT*** *seja excluído antes de reenviar* ***JCL3***

Você já realizou muitas coisas e, com sorte, entende a necessidade de seguir os detalhes. Muito bem!

Bom trabalho - vamos recapitular	A seguir .
JCL pode parecer um pouco complicado e talvez até um pouco desnecessário no início. Não estamos acostumados a usar código para iniciar programas, geralmente clicamos duas vezes neles e eles são executados! No entanto, quando você começa a entrar nos tipos de aplicativos que mantêm os sistemas Z ocupados 24 horas por dia, 7 dias por semana, começa a apreciar a precisão e o poder que a estrutura oferece. Basta dizer que JCL é uma habilidade necessária para qualquer verdadeiro profissional de Z.	Consulte DfSMS Utilities para conhecer outros "utilitários" comuns usados em JCL para gerenciar conjuntos de dados e outros sistemas de armazenamento. Descubra o ambiente Unix dentro do zOS - Unix System Services (USS)