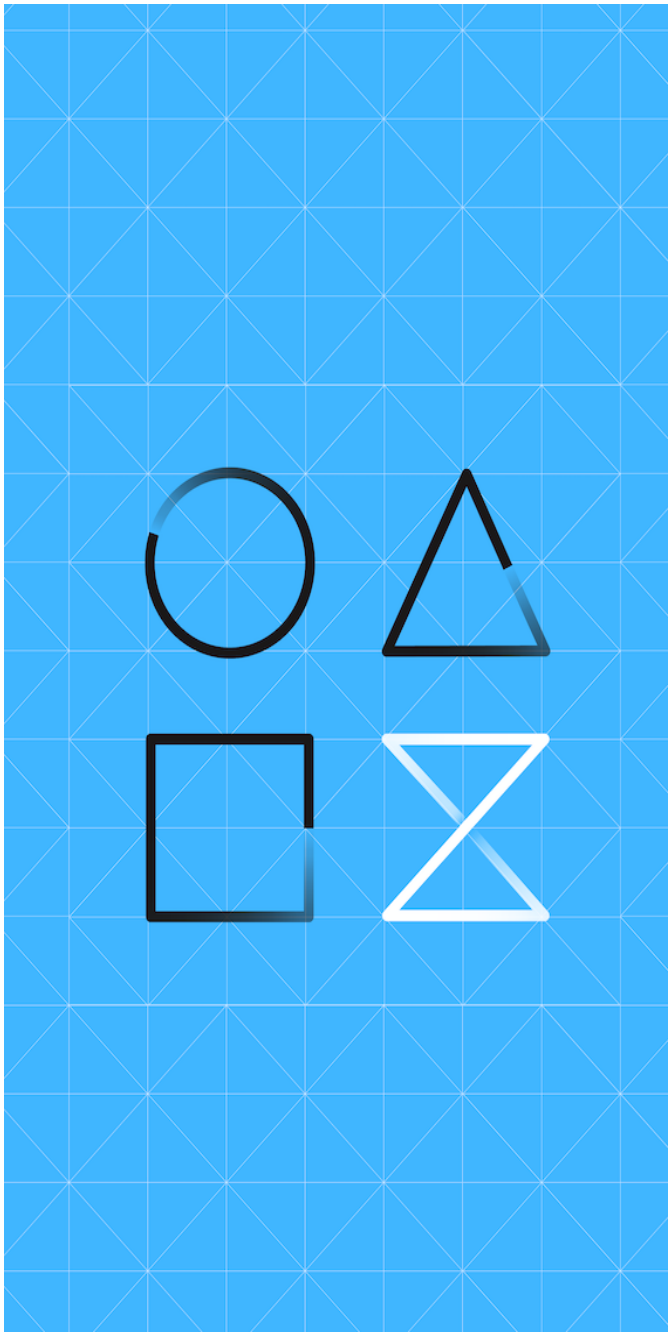


Fundamentals

250117-1355

JCL1

Orchestrating the Enterprise

Automated Translation using IBM Globalization Service

250117-1355

JCL1 - MAKING THINGS HAPPEN

Le Défi

Vous avez déjà vu quelques pages JCL, mais nous n'avons pas vraiment approfondi la question.

Dans ces défis, vous en apprendrez un peu plus sur l'utilisation de JCL, sur son importance dans un environnement Z et sur la manière d'approfondir ces compétences.

JCL est un élément essentiel pour faire avancer les choses dans z/OS et le fait d'être à l'aise avec les concepts et la syntaxe vous permettra de relever les nombreux défis auxquels vous pourriez être confronté au cours de vos explorations.

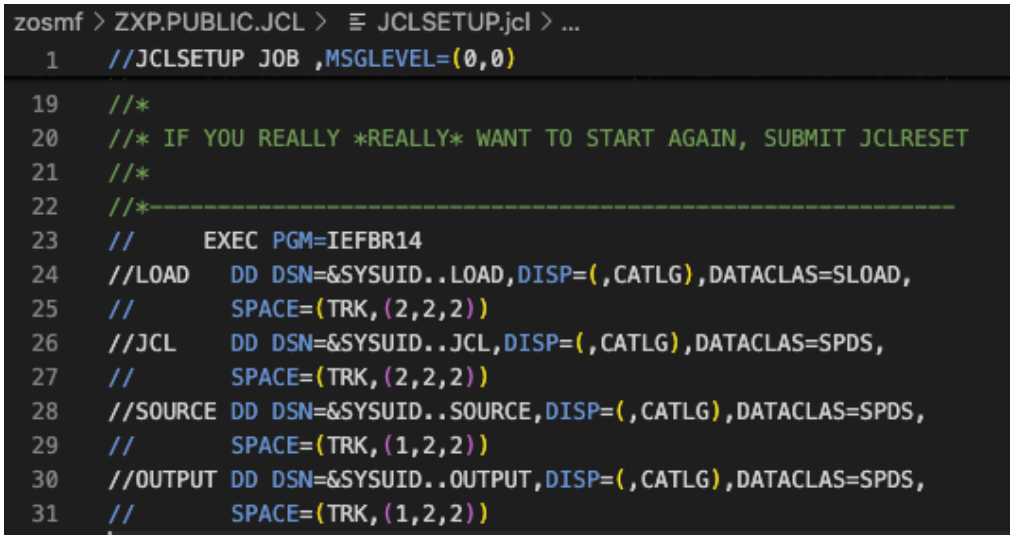
Avant De Commencer

Vous devriez avoir terminé le défi FILES1, qui porte sur les ensembles de données et les membres. Si vous avez compris ces concepts, vous êtes prêts à poursuivre les étapes de ce défi.

Investissement

Étapes	La durée
17	60 minutes

1 LOAD IT UP



Recherchez dans **ZXP.PUBLIC.JCL** un membre nommé **JCLSETUP**.

Il s'agit d'une tâche assez simple qui allouera quelques nouveaux ensembles de données dont vous avez besoin pour ce défi et d'autres.

(La ligne 26 de la capture d'écran crée votre propre ensemble de données JCL)

Vous pouvez voir à la ligne 24 qu'il est fait mention de &SYSUID..LOAD

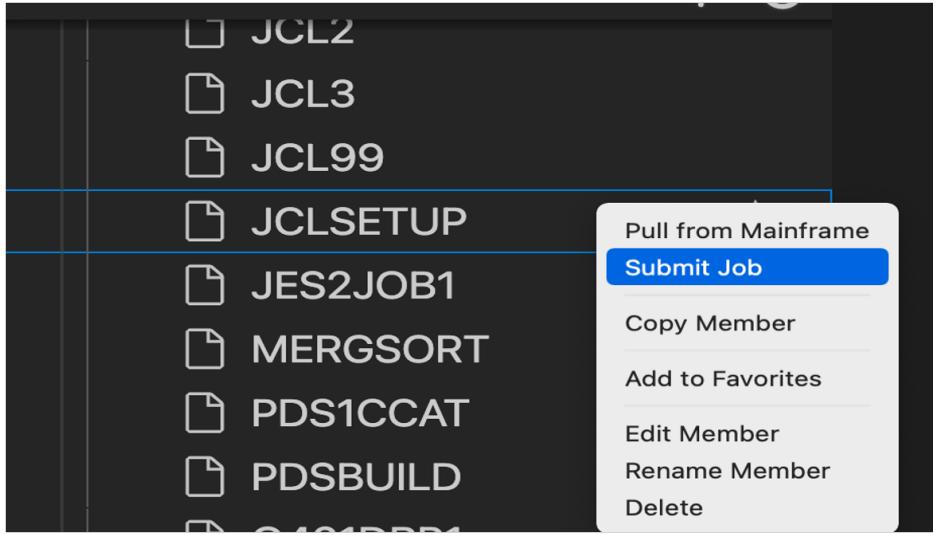
L'esperluette (&) suivie de SYSUID est ce que l'on appelle un "symbole", et lorsque le système le voit, il remplace automatiquement **&SYSUID.** par votre Z-userid.

Notez le "." à la fin - ceci marque la fin du nom symbolique.

Vous n'avez pas besoin de modifier ce travail pour qu'il fonctionne.

Cela signifie que tout le monde peut utiliser le même travail, et qu'il remplacera automatiquement &SYSUID. par *leur* z-userid. Comme c'est pratique !

2 SUBMIT IT



JCL1I250117-1355

Cliquez avec le bouton droit de la souris sur ce travail et sélectionnez Submit Job.

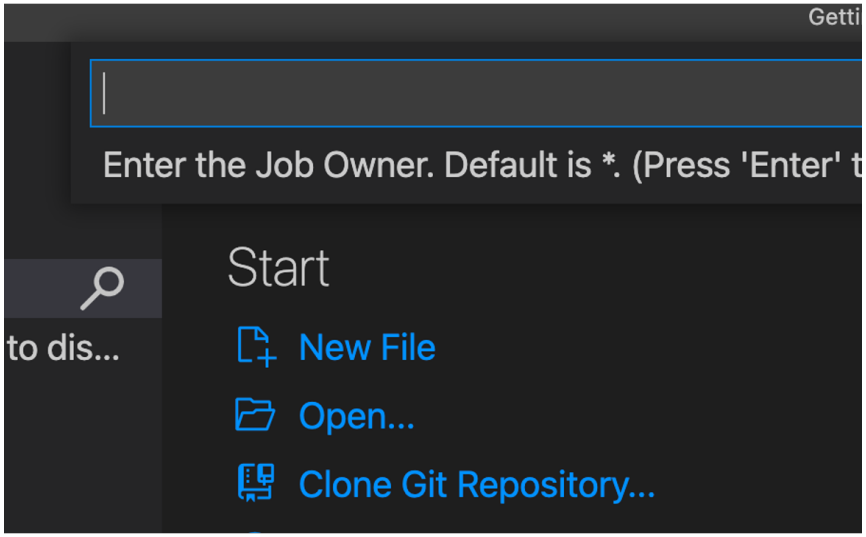
Une remarque sur le mot "Job" : Le JCL est utilisé pour décrire au système exactement ce que vous voulez qu'il fasse.

La tâche que nous confions au système est appelée "Job", et le composant de z/OS qui accepte, traite et gère les résultats de ces jobs est connu sous le nom de **Job Entry Subsystem (JES)**.

Ainsi, pour ce défi, vous avez envoyé un travail à JES pour qu'il traite la tâche que vous venez d'examiner.

Remarque : cette tâche est destinée à créer des ensembles de données pour vous, et elle suppose que vous n'avez pas déjà ces ensembles de données ; *si vous l'exécutez plus d'une fois, vous risquez de voir apparaître des erreurs concernant des noms d'ensembles de données* ***DUPLICATA.***

3 FILTER AND FIND



Vous devriez déjà avoir un profil sous **JOBS** sur le côté gauche de VSCode.

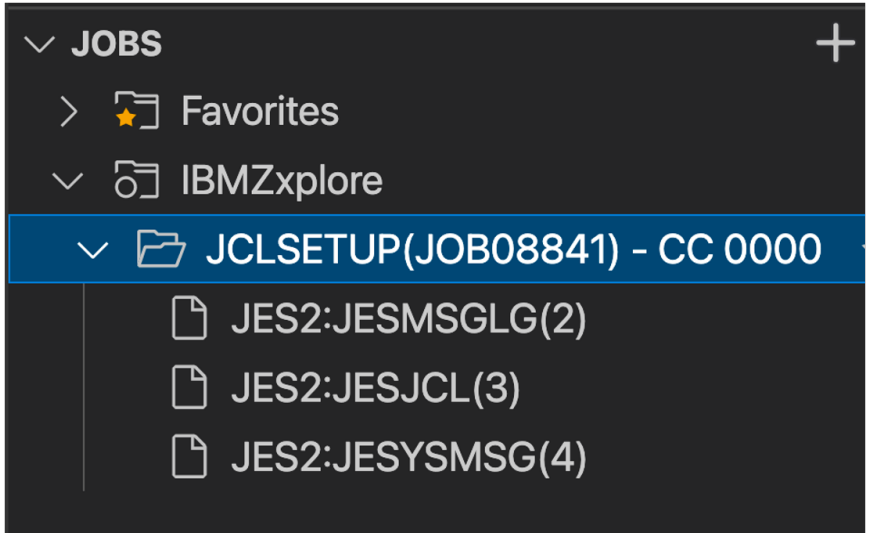
Cliquez sur la loupe ( ) à droite de celle-ci.

- Saisissez votre nom d'utilisateur pour le propriétaire de l'emploi
- Saisir un astérisque (*) pour le préfixe de l'emploi
- Appuyez sur Entrée (blanc, pas de données) pour la recherche de l'identité de l'emploi.

Vous pouvez refaire ce filtre en cliquant à nouveau sur la loupe et en sélectionnant Propriétaire/Préfixe ou Identité du travail. Vous devriez pouvoir y trouver l'emploi que vous venez de soumettre. Rechercher **JCLSETUP**.

L'étape suivante abordera cette question plus en détail.

4 YOU GOT A ZERO. PERFECT!

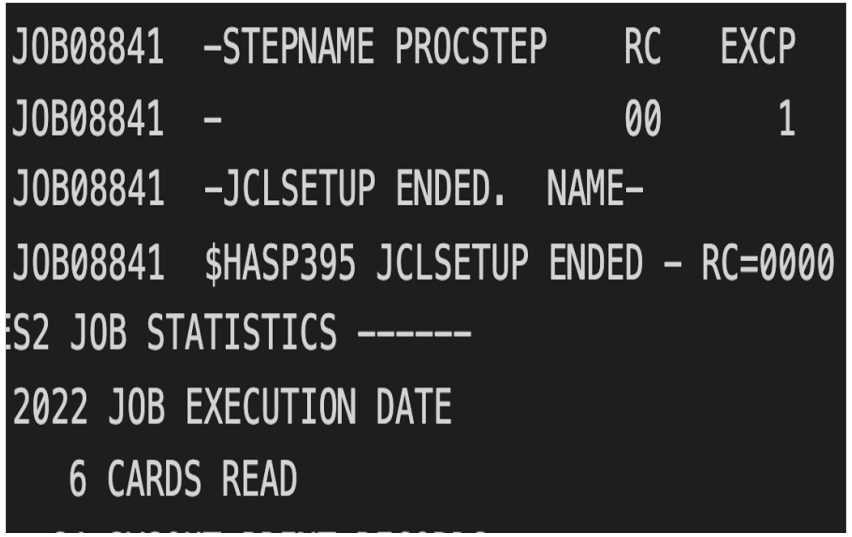


Ouvrez le triangle "twistie" à côté du travail JCLSETUP que vous venez de soumettre. Il y aura probablement d'autres postes à pourvoir, mais vous recherchez plus particulièrement **JCLSETUP**. Si vous l'avez soumis plus d'une fois, trouvez celui qui contient CC 0000 à droite.

Vous verrez également ce numéro dans le **JESMSGLG** une fois que vous aurez ouvert la twistie. Un code de condition (CC) de zéro signifie que tout s'est déroulé comme prévu, sans erreur, donc c'est bon !

Si vous avez obtenu un autre chiffre pour le code d'achèvement, il y a généralement quelque chose qui mérite d'être examiné - et probablement corrigé.

5 JUMP RIGHT TO IT



JCL11250117-1355

Vous avez peut-être remarqué qu'après avoir soumis JCL dans VSCode, un petit message apparaît dans le coin inférieur droit de la fenêtre VSCode.

Plutôt que de fouiller dans le contenu de votre site JOBS, il vous suffit généralement de cliquer sur ce message pour accéder directement à la sortie.

Un travail démarre dans **ACTIVE** pendant son exécution.

Vous pouvez actualiser le statut d'un travail en fermant puis en rouvrant le bouton situé à gauche du nom du travail.

POURQUOI LES JES ET LES JCL SONT-ILS IMPORTANTS ? POURQUOI NE PUIS-JE PAS SIMPLEMENT EXÉCUTER DES PROGRAMMES ?

Lorsque vous soumettez JCL, il est transmis au sous-système d'entrée en service (abrégé en JES).

JES examine le site JCL que vous avez soumis et rassemble toutes les ressources nécessaires à l'accomplissement de la tâche. Sur un système très chargé, il peut être nécessaire de donner à certains travaux une priorité inférieure ou supérieure à d'autres afin que les tâches importantes soient effectuées plus rapidement.

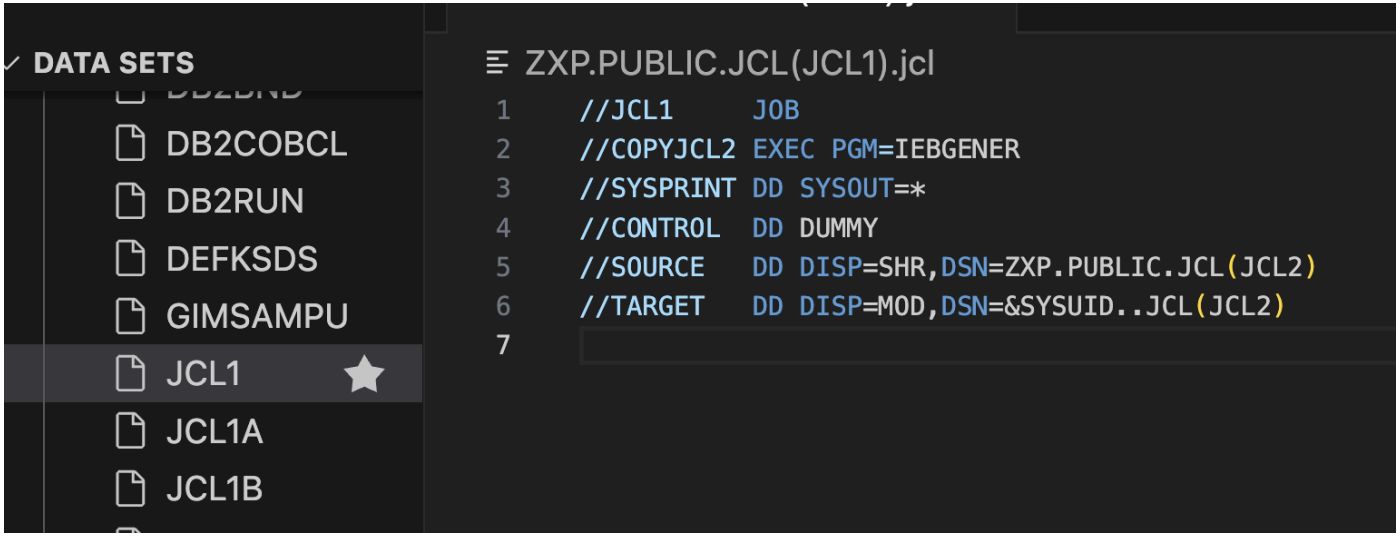
Considérez JCL comme la commande qu'un serveur rédige, et JES comme le personnel de cuisine qui examine la commande et décide de la manière dont il va la traiter.

Le L de JCL signifie "langage", mais il ne s'agit pas vraiment d'un langage de programmation, mais plutôt d'un moyen de décrire efficacement des tâches au système.

Tout ce qui apparaît dans la sortie du travail (le "journal du travail") est une information sur la façon dont le travail s'est déroulé. Comme vous pouvez le constater, il y a beaucoup d'informations ici.

6 MY FIRST COPY JOB

Vous avez maintenant d'autres ensembles de données avec lesquels travailler, et comme il s'agit d'un défi JCL, vous devez obtenir quelques-uns de vos propres JCL pour travailler.



JCL1I250117-1355

Copier le **JCL1** de **ZXP.PUBLIC.JCL** dans votre propre ensemble de données **JCL** puis ouvrez votre copie. Vous *devez* l'avoir dans votre propre jeu de données JCL car vous le modifierez au cours des prochaines étapes.

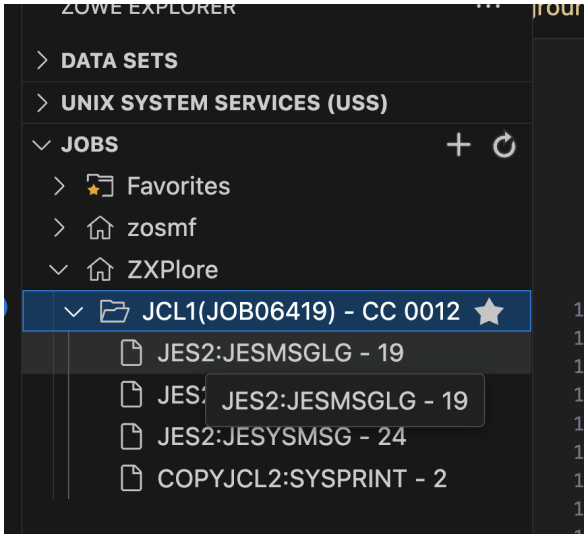
Il se peut que vous deviez fermer et rouvrir votre triangle **DATA SETS** pour rafraîchir la vue, afin que votre JCL1 apparaisse.

Ce travail est fourni pour que vous puissiez copier le membre *JCL2* membre de ZXP.PUBLIC.JCL dans votre jeu de données JCL, mais en utilisant le programme utilitaire au lieu de **IEBGENER** au lieu de VSCode.

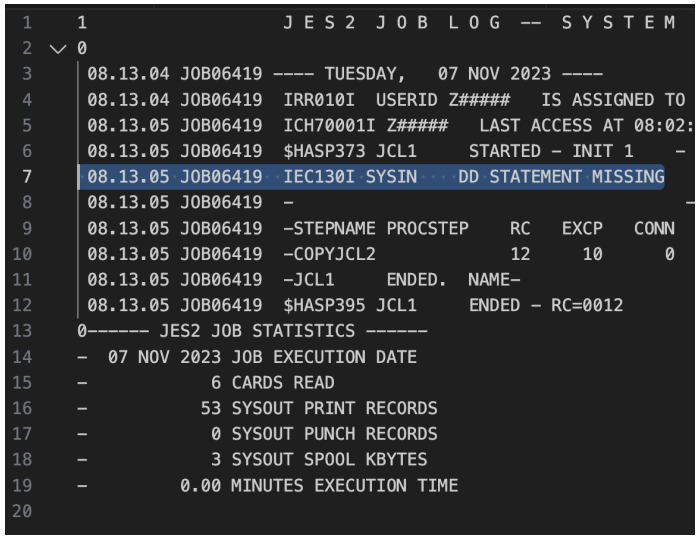
Soumettez le JCL1 de la même manière que vous avez soumis le JCLSETUP, et regardez le journal des tâches.

7 FIRST ERROR

Lorsque vous regardez la section JOBS et que vous voyez votre travail JCL1, vous devriez voir tout de suite que le code d'achèvement est **0012** ; supposez que cela signifie que quelque chose s'est mal passé.



Ouvrez la section **JES2:JESMSGLG** du travail et vérifiez si la cause de l'échec est indiquée.



Dans ce cas, vous pouvez voir que le problème est dû à l'absence de l'instruction DD pour **SYSIN**

Dans JCL, les instructions qui définissent la provenance ou la destination des données sont appelées instructions de **définition des données** ou, plus simplement, "instructions DD ".

Dans ce travail, il n'y a que deux étapes - les deux exécutent le programme **IEBGENER** si vous faites une recherche en ligne, vous trouverez les informations suivantes sur l'installation de IEBGENER :

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

Fichier/DD	Objectif
SYSPRINT	Définit un ensemble de données séquentielles pour les messages. L'ensemble des données peut être écrit sur un périphérique de sortie du système, un volume de bande ou un volume DASD
SYSUT1	Définit l'ensemble des données d'entrée. La "source". Il peut faire référence à un ensemble de données séquentielles existant, à un membre d'un ensemble de données partitionné ou d'un PDSE ou à un fichier UNIX z/OS. Un ensemble de données séquentielles peut être un format de base, un grand format, un format étendu, un

Fichier/DD	Objectif
	format compressé, une entrée spoulée, une bande, un lecteur de cartes, un terminal TSO ou un DUMMY. Notez que **la source doit exister** - si ce n'est pas le cas, IEBGENER se terminera par une erreur **013**.
SYSUT2	Définit l'ensemble des données de sortie. La "cible". Il peut définir un ensemble de données séquentielles, un membre d'un ensemble de données partitionné ou PDSE, un ensemble de données partitionné ou PDSE ou un fichier UNIX z/OS. Un ensemble de données séquentielles peut être un format de base, un grand format, un format étendu, un format compressé, une sortie spoulée, une bande, une perforation de carte, une imprimante, un terminal TSO ou un DUMMY
SYSIN	Définit l'ensemble des données de contrôle ou spécifie DUMMY lorsque la sortie est séquentielle et qu'aucune édition n'est spécifiée. L'ensemble de données de contrôle réside normalement dans le flux d'entrée ; toutefois, il peut être défini comme un membre d'un ensemble de données partitionné ou d'un PDSE.

Si tout va bien, vous pouvez constater que les fichiers attribués à IEBGENER dans la tâche JCL1, à l'aide des instructions DD, ne correspondent pas aux fichiers **requis par** le programme.

Apportez les modifications nécessaires aux instructions DD dans votre membre JCL1**(n'oubliez pas de sauvegarder !)** et soumettre à nouveau.

Si vous avez effectué les modifications correctes, le travail devrait s'achever avec CC=0000.

Si ce n'est pas le cas, vérifiez à nouveau le journal des travaux pour y trouver des messages indiquant la cause des nouvelles erreurs ; effectuez les ajustements nécessaires et réessayez.

Avec ce type de programme, il est facile d'identifier et de corriger ce qui n'a pas fonctionné, par exemple des déclarations de DD incorrectes ou manquantes - c'est dans le livre !

Une fois le travail JCL1 terminé avec succès, vous devriez avoir un nouveau membre dans votre jeu de données **JCL2** dans votre jeu de données JCL.

(Vous aurez également un **JCL3** qui est créé par la deuxième étape du travail 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
```


Modifiez votre copie personnelle de la **JCL2** définition du poste.

Il se peut que vous deviez fermer et rouvrir votre triangle **DATA SETS** pour rafraîchir la vue, afin que votre JCL2 apparaisse.

Ce JCL est utilisé pour compiler et exécuter du code COBOL. Après la compilation, le programme résultant sera placé dans votre ensemble de données **LOAD** ensemble de données.

Remarque : les programmes de l'ensemble de données LOAD sont binaires - vous ne pourrez pas les visualiser avec VSCode.

Recherchez la ligne qui commence par **//COBRUN** - c'est le début d'une "étape" du travail qui exécutera le compilateur COBOL.

Sur la ligne suivante, vous pouvez voir l'ensemble des données d'entrée (la source) à la ligne 18 (**//COBOL.SYSIN**), et l'emplacement de la sortie sur la ligne suivante (**//LKED.SYSLMOD**).

Les lignes qui suivent le début de l'étape **//RUN** ont un format similaire :

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

- "ddname" est également appelé nom de fichier - le nom utilisé par les programmes pour accéder aux données dans les ensembles de données
- le "dataset" est l'emplacement réel des données - il peut changer, mais le programme n'a pas besoin de le savoir
- "l'accès (ou la disposition) indique comment le programme peut utiliser l'ensemble de données

Si l'étape de compilation se termine avec un code de complétion de 0 (parce qu'il n'y a pas eu de problème), le jobtep exécutera le programme **CBL0001** programme.

Tout s'explique jusqu'à présent ? Vous utiliserez JCL pour compiler le code source de COBOL, puis pour exécuter le programme résultant.

QUE SIGNIFIE COMPILER ? QU'EST-CE QUE LE COBOL ?

COBOL est un langage de programmation utilisé dans de nombreuses institutions financières, médicales et gouvernementales. Son haut degré de précision mathématique et ses méthodes de codage directes en font une solution naturelle lorsque les programmes doivent être rapides, précis et faciles à comprendre.

Le code écrit par les humains doit être transformé en code machine pour pouvoir être exécuté en tant que programme. La compilation est une étape qui permet d'effectuer cette transformation. Notre site JCL comporte deux étapes principales : la compilation du code source en code machine et l'exécution du programme.

Ce programme particulier nécessite également deux fichiers d'entrée et écrit dans un fichier de sortie. Nous spécifierons donc également ces fichiers (ensembles de données) dans le site JCL.

9 RUN, CODE, RUN

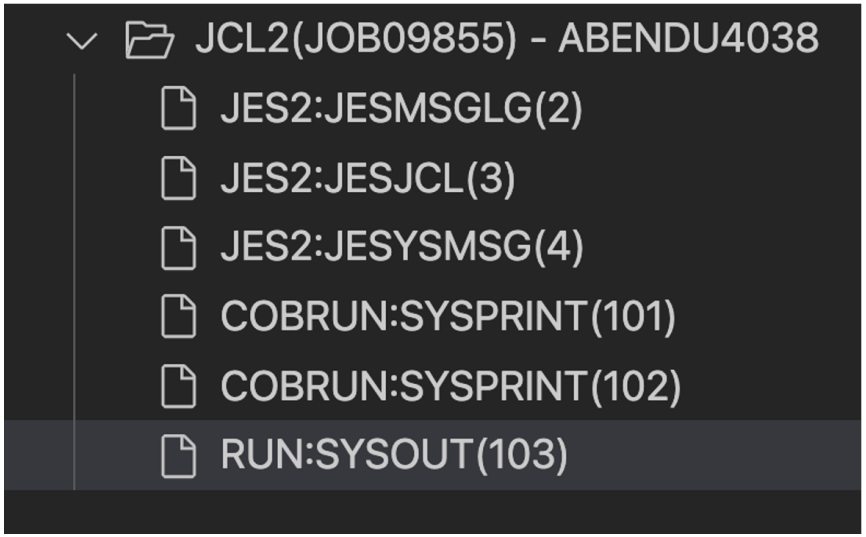
Après avoir compilé avec succès le code COBOL, JES exécutera le programme (la commande **//RUN EXEC PGM=CBL0001**) et lui indiquera où trouver les ensembles de données d'entrée, ainsi que l'endroit où la sortie sera stockée dans votre ensemble de données de sortie (OUTPUT)

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

Le nom de l'instruction DD est ce qui vient directement après les doubles barres obliques, par exemple " FNAMES " et " LNAMES ".

Toutes les lignes commençant par **//*** sont des commentaires et sont ignorées par JES. Les lignes commentées sont utiles pour fournir des informations ou pour conserver des lignes de code que nous pourrions utiliser plus tard, mais dont nous n'avons pas besoin pour l'instant.

10 THEY CAN'T ALL BE ZEROES



Soumettre **JCL2** à partir de votre ensemble de données JCL, puis examinez la sortie, en utilisant ce que vous avez appris lors des étapes précédentes de ce défi.

REMARQUE : *Vous* obtiendrez un **ABEND** (abréviation de Abnormal End), ce qui signifie que quelque chose n'est pas encore tout à fait correct.

Mais ne vous inquiétez pas : grâce à vos nouvelles compétences, vous parviendrez à tirer les choses au clair !

Dans les étapes précédentes du travail JCL1, vous pouviez utiliser la documentation de **IEBGENER** pour déterminer les instructions DD *nécessaires* au fonctionnement du programme ; dans le cas présent, il n'y a pas de documentation, mais seulement le code COBOL lui-même.

Dans l'étape suivante, vous examinerez le code COBOL et verrez comment le code réel correspond au code JCL utilisé pour le compiler et l'exécuter.

Et encore une fois, ne vous inquiétez pas ! Vous n'avez pas besoin de devenir un expert de COBOL pour résoudre ce problème - rappelez-vous qu'il s'agit d'un défi, et non pas d'un défi de **JCL** d'un défi, pas d'un défi COBOL.

Pour que les programmes z/OS puissent travailler avec des ensembles de données dans le cadre d'un travail, quatre conditions doivent être remplies :

1. la définition interne du fichier qui représente la structure des données et le contenu que le programme créera, lira, modifiera ou supprimera - elle est définie à l'intérieur du programme
2. l'ensemble de données actuel qu'une exécution particulière du programme peut utiliser - chaque fois que le programme est exécuté, il peut utiliser différents ensembles de données, à condition qu'ils aient le format correct attendu par le programme ; c'est le nom utilisé dans le paramètre **DSN=** d'une instruction DD
3. un choix correct de la disposition de l'ensemble de données pour indiquer s'il doit être créé, supprimé, transmis - il s'agit du paramètre ** `DISP=*` * d'une déclaration DD
4. une déclaration DD qui lie le fichier interne du programme à l'ensemble de données particulier - cette déclaration doit correspondre au nom de la définition du fichier du programme et spécifier l'ensemble de données requis ainsi que la disposition

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.
```


L'instruction JCL commençant par **//COBOL.SYSIN** pointe vers l'ensemble de données du code source COBOL qu'il va compiler, il faut donc commencer par là. Ouvrez le code source de ce programme à l'adresse VSCode, et commencez par examiner la zone **FILE-CONTROL** zone.

C'est ici que vous obtenez les noms utilisés par le programme que vous devez faire correspondre dans le JCL. Par exemple, **FIRST-NAME** est une référence d'enregistrement dans le code COBOL pour un fichier, et qui est assigné (ou lié) à l'instruction **FNAMES** DD dans le JCL.

Ouvrez JCL, COBOL code, et le joblog - regardez la sortie, et vous devriez être en mesure de voir ce qu'il faut faire

changement très simple d' ***une seule lettre***

doit être apportée à la tâche **JCL2** pour que tout soit lié entre le programme COBOL, les ensembles de données et le JCL.

Lorsque vous aurez résolu le problème dans JCL2, il devrait s'exécuter avec CC=0000 et vous *trouverez la sortie correcte dans le membre correct* de votre ensemble de données 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
```

Ouvrez le membre **JCL3** de votre jeu de données JCL et regardez ce qu'il contient. Vous verrez que ce JCL contient un certain nombre d'étapes, chacune d'entre elles utilisant le programme utilitaire pour diriger un ou plusieurs enregistrements dans un ensemble de données séquentiel **IEBGENER** pour diriger un ou plusieurs enregistrements dans un jeu de données séquentiel.

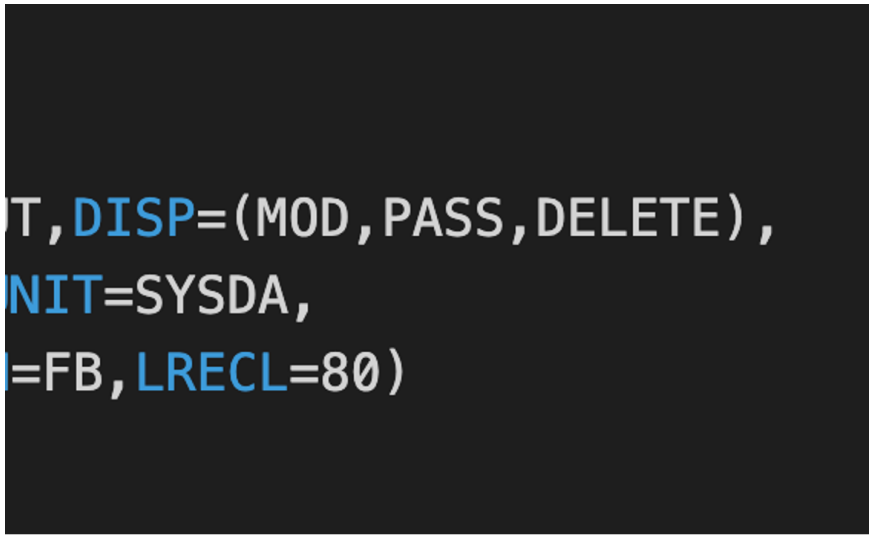
Il s'agit d'une extension assez simple de la tâche JCL1 que vous avez étudiée précédemment - la grande différence étant la façon dont les données "source" sont définies.

Dans ce travail, les données pour toutes les définitions de données sont définies comme étant "in-stream" **SYSUT1** sont définies comme "in-stream" - elles se trouvent juste là dans le travail après l'instruction DD. Il est spécifié par l'option **DD *** au lieu d'un paramètre **DSN=** paramètre. Le programme IEBGENER copiera les données de SYSUT1 jusqu'à la prochaine instruction JCL commençant par **//** ou **/***

Vous verrez qu'il y a une ligne d'en-tête, quelques lignes avec des informations sur les gares pour les arrêts de pointe entre Poughkeepsie, NY et Grand Central Terminal à NYC, suivies d'un texte sur les heures d'exploitation.

Cela semble assez simple... quelle sera la partie la plus délicate ?

13 A FRIENDLY DISPOSITION



Dans tous les JCL que vous avez utilisés jusqu'à présent, vous avez rencontré une sorte de paramètre (de disposition) **DISP=** (disposition).

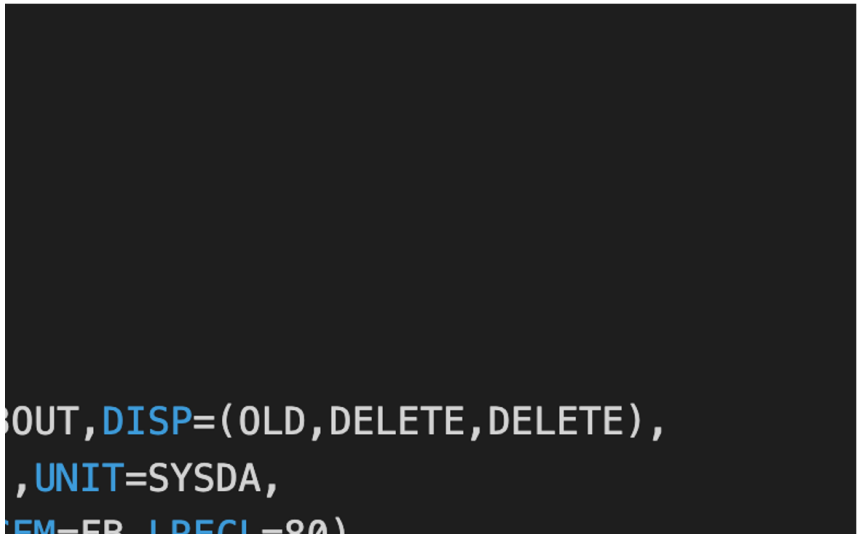
DISP sont utilisés pour décrire comment JCL doit utiliser ou créer un ensemble de données, et ce qu'il faut en faire une fois le travail, ou l'étape du travail, terminé.

Un paramètre standard de DISP comporte trois parties. Le premier paramètre est le statut, qui peut être l'un des suivants :

Paramètres	Signification
NOUVEAU	Créer un nouvel ensemble de données
SHR	Réutiliser un ensemble de données existant et permettre à d'autres personnes de l'utiliser si elles le souhaitent
ANCIEN	Réutiliser un ensemble de données existant, mais ne pas laisser d'autres personnes l'utiliser pendant que nous l'utilisons

Paramètres	Signification
MOD	Pour les ensembles de données séquentielles uniquement. Réutiliser un ensemble de données existant, mais n'ajouter que les nouveaux enregistrements au bas de l'ensemble. Si aucun ensemble de données n'existe, créez-en un nouveau.

14 WHAT'S YOUR STATUS?



Le deuxième champ du paramètre DISP décrit ce qui doit arriver à l'ensemble de données dans le cas d'un achèvement normal de l'étape de travail, et le troisième champ indique ce qui doit arriver à l'ensemble de données dans le cas d'un échec de l'étape de travail.

Il existe un certain nombre de valeurs qui peuvent être utilisées ici, mais pour ce défi, vous n'avez besoin que des valeurs suivantes :

Field2	Signification
DELETE	L'effacer complètement de la mémoire
CATLG	Enregistrez l'ensemble des données afin de pouvoir les utiliser une fois le travail terminé
PASSER	Une fois cette étape terminée, conservez-la pour que les étapes suivantes (dans le même travail) puissent l'utiliser

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
********************************************
```


Vous devriez remarquer de nombreuses mentions de **DUMMY** dans les déclarations de DD.

Ne vous inquiétez pas, ce JCL n'insulte personne ; c'est juste une façon de dire "Cette allocation de fichier est nécessaire, mais cette fois-ci, elle ne sera pas utilisée pour quoi que ce soit, donc cela n'a pas d'importance".

Comme vous l'avez vu précédemment, le programme **IEBGENER** programme s'exécutant à chaque étape nécessite :

- **SYSIN** une déclaration de fichier d'entrée (DD) pour les commandes de contrôle
- **SYSPRINT** une déclaration DD de sortie pour le programme afin de signaler le succès, l'échec, la progression
- **SYSUT1** une déclaration DD "source" à partir de laquelle les données doivent être copiées
- **SYSUT2** une déclaration DD "cible" dans laquelle les données de SYSUT1 doivent être copiées

Mais dans ce travail, la sortie du programme n'est pas nécessaire, et il n'y a pas d'instructions de contrôle requises, donc DUMMY est une façon de dire "Cela n'a pas d'importance, ne perdez pas votre temps à configurer cela" - prenez simplement l'action par défaut et copiez l'ensemble de données associé à **SYSUT1** dans le jeu de données associé à **SYSUT2**.

16 RIGHT ON TIME

Soumettez votre copie du travail **JCL3** et regardez le résultat.

NOTE : vous devez vous attendre à ce que ce travail se termine par le code 0000 - cela signifie que rien n'est cassé - cela ne signifie pas que la sortie est correcte !

Vous devez procéder à deux vérifications pour que ce travail s'exécute correctement à 100 % :

- une liste complète des **10 gares**, de Poughkeepsie à Grand Central Terminal
- les informations en haut et en bas de la série de données.
- Vous ne devriez ***pas avoir d'arrêts répétés***. Si Beacon ou Cortlandt y figure deux fois, c'est qu'il y a encore quelque chose à corriger.

REMARQUE : *Vous devrez supprimer l'ensemble de données de sortie* ***JCL3OUT*** *à chaque fois avant de le soumettre à nouveau* ***JCL3***

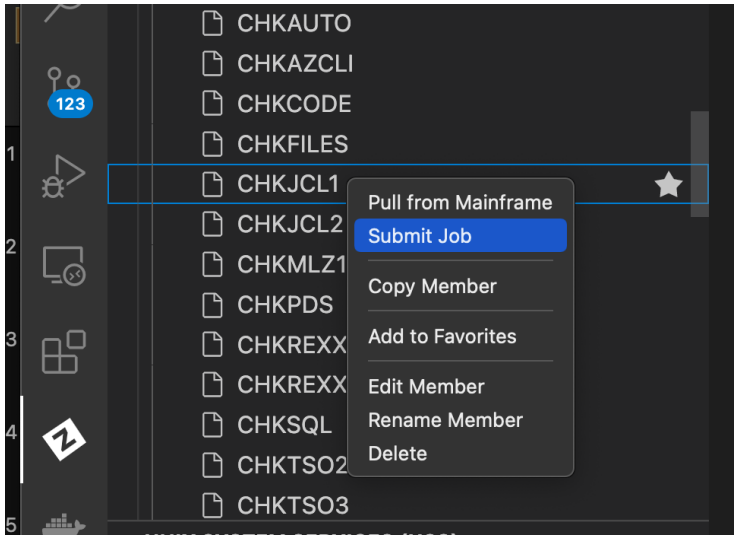
Si vous le souhaitez, consultez le JCL des tâches JCLSETUP et PDSBUILD pour voir comment vous pourriez supprimer automatiquement JCL3OUT en ajoutant une étape supplémentaire au début de JCL3.

Soumettez à nouveau le site JCL3 et vérifiez si vous obtenez le résultat correct.

Une fois cette opération terminée, vous devriez obtenir la sortie complète dans votre ensemble de données séquentielles, totalisant 23 lignes (enregistrements) **JCL3OUT** séquentiel, totalisant 23 lignes (enregistrements), ce qui correspond à la vue suivante :

```
*****************************************
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?



Soumettez maintenant le travail **CHKJCL1** à partir de **ZXP.PUBLIC.JCL** pour valider la sortie correcte de JCL2 et JCL3, et espérer voir le code d'achèvement (CC) de 0000.

Si CHKJCL1 renvoie CC=0127, revenez en arrière, vérifiez et ajustez votre travail pour JCL2 et JCL3, et soumettez à nouveau ces travaux si nécessaire.

Vérifiez à nouveau que la sortie est correcte en soumettant à nouveau **CHKJCL1** jusqu'à ce que vous obteniez CC=0000.

ENCORE : *Vous devrez vous assurer que le jeu de données de sortie* ***JCL3OUT*** *est supprimé avant de soumettre à nouveau le* ***JCL3***

Vous avez accompli beaucoup de choses et vous comprenez, je l'espère, la nécessité de suivre les détails . Bravo !

Bon travail - récapitulons	A suivre .
JCL peut sembler un peu difficile, voire inutile au début. Nous n'avons pas l'habitude d'utiliser du code pour lancer des programmes, il suffit généralement de double-cliquer dessus pour qu'ils s'exécutent ! Cependant, lorsque vous commencez à vous intéresser aux types d'applications qui occupent les systèmes Z 24 heures sur 24 et 7 jours sur 7, vous commencez à apprécier la précision et la puissance que la structure vous offre. Il suffit de dire que JCL est une compétence nécessaire pour tout véritable professionnel de la Z.	Consultez DfSMS Utilities pour d'autres "utilitaires" courants utilisés dans JCL pour gérer les ensembles de données et d'autres systèmes de stockage. Découvrez l'environnement Unix dans zOS - Unix System Services (USS)