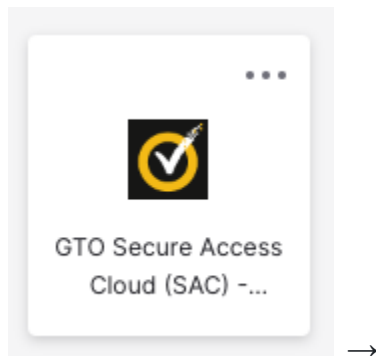


# Team Build Workshop Scenario

## Getting Started

1. Login to the workshop system using the given URL, username, and password, and follow the steps your instructor provides



### APPLICATIONS



MSD-GCP-ZDNT-



MSD-GCP-ZDNT-



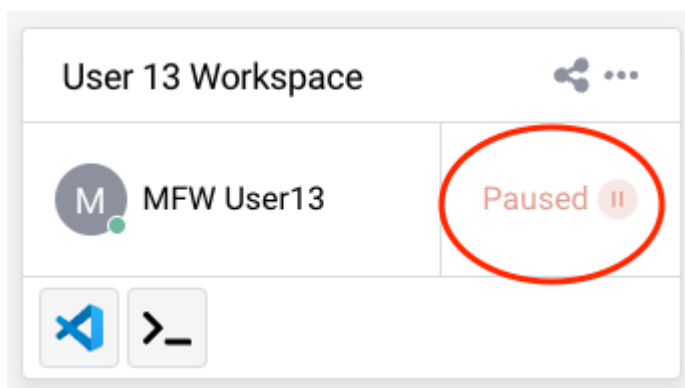
MSD-Workshop



MSD-Workshop-2

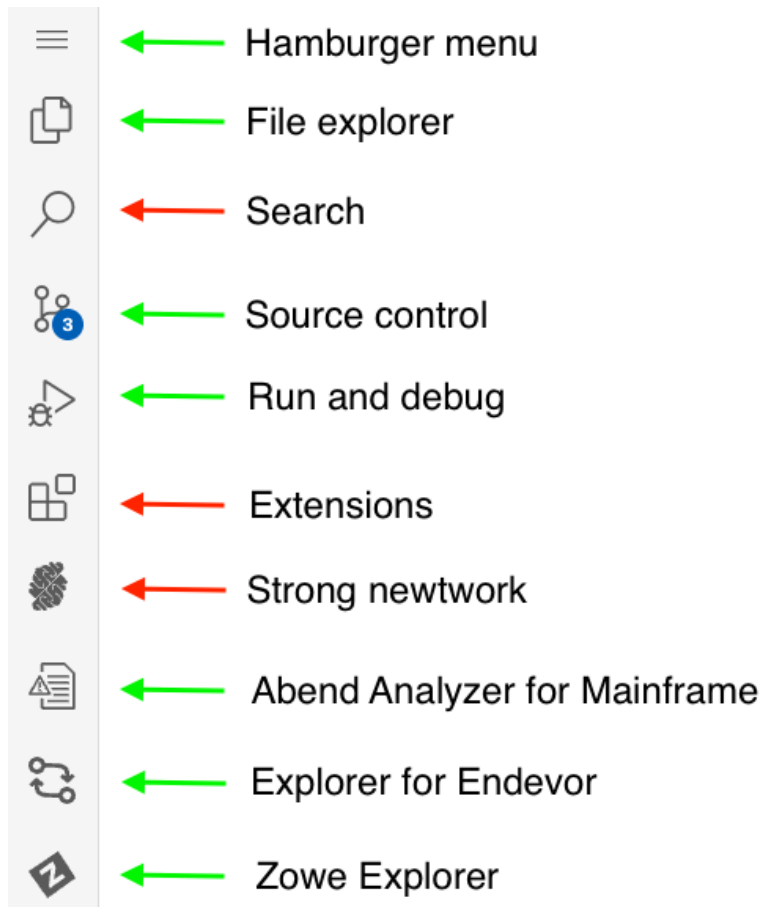


MSD-Workshop-3



2. You are in the secure cloud environment which runs VS Code and is connected to the Mainframe
3. Make sure the initial build process has been completed successfully (exit code: 0 message in the active terminal)
4. Close the terminal from it's right top corner

## Get familiar with the VSCode Activity Bar

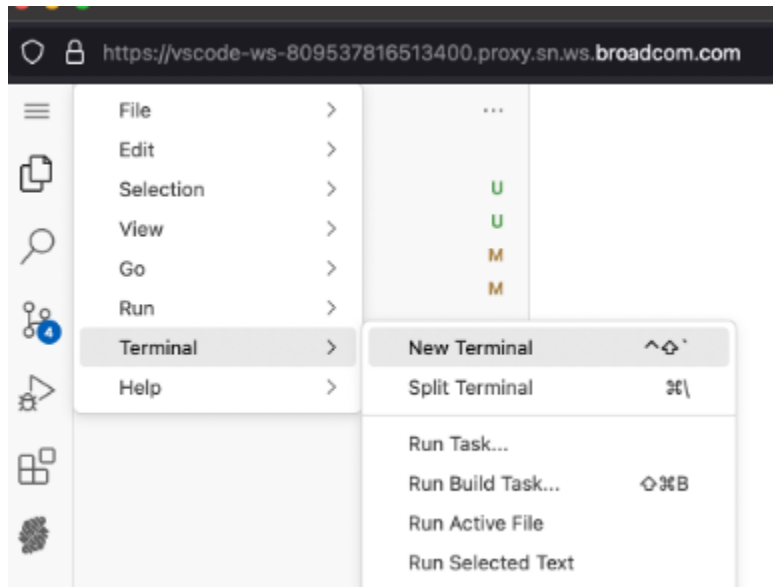


# Scenario 1: Exporting a Build Script from Endeavor

## Step 1:

Open a terminal window by selecting the three horizontal lines on the top left (Burger Icon) -> Terminal -> New Terminal

Reference screenshot:



## Step 2:

In this next step, we are going to take the existing logic for our application which is currently part of the Endeavor processor automation, and use it to create a Team Build project. We do this by using the exportz command to connect to the Endeavor web services for the specified environment, entry stage, system and subsystem to extract the code and the existing build logic. Since Endeavor processors build into Endeavor managed output libraries, we will give a high level qualifier that we own to map the build output to (--dataset-hlq). Copy paste the below command in the terminal

Unset

```
./exportz --environment DEV --system DOGGOS --subsystem  
CUST0## --sn 1 --base-url  
http://10.1.2.120:6002/EndeavorService/api/v2 --instance  
ENDEVOR --user cust0## --dataset-hlq 'CUST0##.BLDZ.DOGGOS'  
--ssh-port 2022
```

### Important Notes:

- a). Replace **##** in the above command with your User ID number.
  - b). The command run will prompt for a password. Enter the Password.
- Both the User ID number and Password will be shared by the instructor.

Here are a couple examples:

#### Example 1:

If you are assigned User 03, then replace **##** with 03 and the command will look like below:

Unset

```
./exportz --environment DEV --system DOGGOS --subsystem  
CUST003 --sn 1 --base-url  
http://10.1.2.120:6002/EndevorService/api/v2 --instance  
ENDEVOR --user cust003 --dataset-hlq 'CUST003.BLDZ.DOGGOS'  
--ssh-port 2022
```

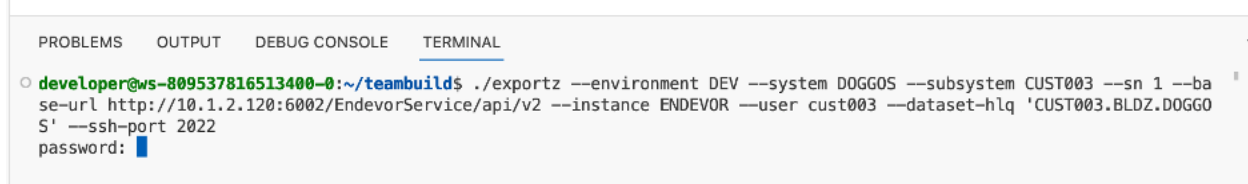
#### Example 2:

If you are assigned User 21, then replace **##** with 21 and the command will look like below:

Unset

```
./exportz --environment DEV --system DOGGOS --subsystem  
CUST021 --sn 1 --base-url  
http://10.1.2.120:6002/EndevorService/api/v2 --instance  
ENDEVOR --user cust021 --dataset-hlq 'CUST021.BLDZ.DOGGOS'  
--ssh-port 2022
```

### Reference screenshots:



The screenshot shows a terminal window with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is active, displaying the command execution and the password prompt. The command is: `./exportz --environment DEV --system DOGGOS --subsystem CUST003 --sn 1 --base-url http://10.1.2.120:6002/EndevorService/api/v2 --instance ENDEVOR --user cust003 --dataset-hlq 'CUST003.BLDZ.DOGGOS' --ssh-port 2022`. The prompt is `password:` followed by a blue cursor.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
developer@ws-809537816513400-0:~/teambuild$ ./exportz --environment DEV --system DOGGOS --subsystem CUST003 --sn 1 --base-url http://10.1.2.120:6002/EndevorService/api/v2 --instance ENDEVOR --user cust003 --dataset-hlq 'CUST003.BLDZ.DOGGOS' --ssh-port 2022
password:
[INFO] Connecting cust003@10.1.2.120:2022...
[INFO] Remote encoding (on mainframe): cp1047
[INFO] Obtaining environment DEV information...
[INFO] Obtaining system DEV/DOGGOS information...
[INFO] Obtaining subsystem DEV/DOGGOS/CUST003 information...
[INFO] Obtaining environment QA information...
[INFO] Obtaining system QA/DOGGOS information...
[INFO] Obtaining subsystem QA/DOGGOS/DOGGOS information...
[INFO] Obtaining environment PRD information...
[INFO] Obtaining system PRD/DOGGOS information...
[INFO] Obtaining subsystem PRD/DOGGOS/DOGGOS information...
[INFO] Obtaining information for type COBOL in DOGGOS...
[INFO] Retrieving element COBOL/DOGGOS03...
[INFO] Retrieving processor PRD/2/EA/PROCESS/PROCESS/GCOBP...
[INFO] Compiling processor EA_GCOBP...
[INFO] Retrieving processor PRD/2/EA/PROCESS/PROCESS/GCOBPD...
[INFO] Compiling processor EA_GCOBPD...
[INFO] Fetching information about dependencies for DEV/1/DOGGOS/CUST003/COBOL/DOGGOS03...
[INFO] Obtaining information for type COBCOPY in DOGGOS...
[INFO] Retrieving element COBCOPY/ADOPTRPT...
[INFO] Retrieving element COBCOPY/DATETIME...
[INFO] Retrieving element COBCOPY/DOGADOPT...
[INFO] Creating BUILDZ.js...
[INFO] Creating dependencies.json...
[INFO] Creating WORKSPACEZ.js...
[INFO] Creating scripts/endevor/element_overrides.json...
[INFO] Creating scripts/endevor/inventory.json...
[INFO] Creating scripts/endevor/maps.json...
[INFO] Creating scripts/endevor/pgroup.json...
[INFO] Creating scripts/endevor/site_symbols.json...
[INFO] Writing exportz_report.log...
```

### Step 3:

Now we want to make sure the exported application builds successfully. We will use the syncz command to create a mirror of our exported code in Unix System Services in z/OS. Afterwards, we request z/OS to invoke the build engine on the directory we've labeled 'src'. This is done by adding the 'after' command to the syncz call via the -a parameter. The --proc 1 option instructs the build to run with a single process. The definition of what user ID to use and where to mirror is done via the \*.yml files that are defined for syncz.

Note that passwords are not needed here because we have set up key based authentication in SSH which is used to connect to the USS.

Copy paste the below command in the terminal

```
./syncz -a "src::bldz --proc 1"
```

Reference screenshots:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
○ developer@ws-809537816513400-0:~/teambuild$ ./exportz --environment DEV --system DOGGOS --subsystem CUST003 --sn 1 --base-url http://10.1.2.120:6002/EndevorService/api/v2 --instance ENDEVOR --user cust003 --dataset-hlq 'CUST003.BLDZ.DOGGOS' --ssh-port 2022
password: █
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
developer@ws-809537816513400-0:~/teambuild$ ./exportz --environment DEV --system DOGGOS --subsystem CUST003 --sn 1 --base-url http://10.1.2.120:6002/EndevorService/api/v2 --instance ENDEVOR --user cust003 --dataset-hlq 'CUST003.BLDZ.DOGGOS' --ssh-port 2022
password:
[INFO] Connecting cust003@10.1.2.120:2022...
[INFO] Remote encoding (on mainframe): cp1047
[INFO] Obtaining environment DEV information...
[INFO] Obtaining system DEV/DOGGOS information...
[INFO] Obtaining subsystem DEV/DOGGOS/CUST003 information...
[INFO] Obtaining environment QA information...
[INFO] Obtaining system QA/DOGGOS information...
[INFO] Obtaining subsystem QA/DOGGOS/DOGGOS information...
[INFO] Obtaining environment PRD information...
[INFO] Obtaining system PRD/DOGGOS information...
[INFO] Obtaining subsystem PRD/DOGGOS/DOGGOS information...
[INFO] Obtaining information for type COBOL in DOGGOS...
[INFO] Retrieving element COBOL/DOGGOS03...
[INFO] Retrieving processor PRD/2/EA/PROCESS/PROCESS/GCOBP...
[INFO] Compiling processor EA_GCOBP...
[INFO] Retrieving processor PRD/2/EA/PROCESS/PROCESS/GCOBPD...
[INFO] Compiling processor EA_GCOBPD...
[INFO] Fetching information about dependencies for DEV/1/DOGGOS/CUST003/COBOL/DOGGOS03...
[INFO] Obtaining information for type COBCOPY in DOGGOS...
[INFO] Retrieving element COBCOPY/ADOPTRPT...
[INFO] Retrieving element COBCOPY/DATETIME...
[INFO] Retrieving element COBCOPY/DOGADOPT...
[INFO] Creating BUILDZ.js...
[INFO] Creating dependencies.json...
[INFO] Creating WORKSPACEZ.js...
[INFO] Creating scripts/endevor/element_overrides.json...
[INFO] Creating scripts/endevor/inventory.json...
[INFO] Creating scripts/endevor/maps.json...
[INFO] Creating scripts/endevor/pgroup.json...
[INFO] Creating scripts/endevor/site_symbols.json...
[INFO] Writing exportz_report.log...
```

## Scenario 2: Creating a Build Script from Scratch

### Step 1:

1. Go back to the Strong Network workspaces window and launch the workspace Team Build Scenario 2
2. Make sure the initial build process has successfully completed. (exit code: 0 message in the active terminal)
3. Close the terminal from it's right top corner

### Step 2:

1. Click on the 3 horizontal lines at the top left of the window (the 'Hamburger' icon)
2. Select File > Open Folder
3. Navigate to /home/developer/teamBuildStd (choose 'teamBuildStd' from the dropdown list that appears and click 'OK')
4. Expand the Explorer view (first icon under the 'Hamburger' icon on the top left)
5. Locate the BUILDZ.js file list treeview and double click to edit it. It should be an empty file. BUILDZ.js is the main build file for Team Build and what the bldz process looks for.
6. First, we need to add in the required Standard Library includes. This includes the functions for building and binding COBOL applications. Add the following 2 lines:

Unset

```
var compile = require("bldz/std/exp/rules/compile")
var binder = require("bldz/std/exp/rules/binder")
```

7. Now we will direct the script to compile all the COBOL in the src folder. Add the following line:

Unset

```
var compile_rules = compile.cobol({ srcs:
"src/*.cbl"})
```

8. The script creates rules, which we can use as a dependency for the binder to create a load module. Add the following line to the script:

Unset

```
var bind_rules = binder.bind({deps:
compile_rules.rules, syslibs: ["//CEE.SCEELKED"]})
```

9. Save the BUILDZ.js file (Hamburger Icon > File > Save)
10. Now we will test our new build script. Open a new terminal window (Hamburger Icon > Terminal > New Terminal)
11. Run the `./syncz -a "src::bldz --proc 1"` command to compile the application. As before, syncz creates a mirror of your working directory on z/OS and then via the `-a` parameter, directs the bldz engine to run on the 'src' directory we identified in the \*.yml files.
12. If all goes well, you should see the compilation output in the terminal with a message indicating all complies succeeded. Congratulations! You have created a simple build script with only 4 lines of code! If you wish to review the listings, they will have synchronized back to your workspace in the listings directory.



Reference screenshot:

