z/OS Connect OpenAPI 3

Designer and z/OS Native server Experiences and Observations

Mitch Johnson
mitchj@us.ibm.com
Washington Systems Center

The significance of OpenAPI Specification*

- The industry standard framework for describing REST APIs

> The OpenAPI Initiative (OAI) was created by a consortium of forward-looking industry experts who recognize the immense value of standardizing on how APIs are described. As an open governance structure under the Linux Foundation, the OAI is focused on creating, evolving and promoting a vendor neutral description format. The OpenAPI Specification was originally based on the Swagger Specification, donated by SmartBear Software.

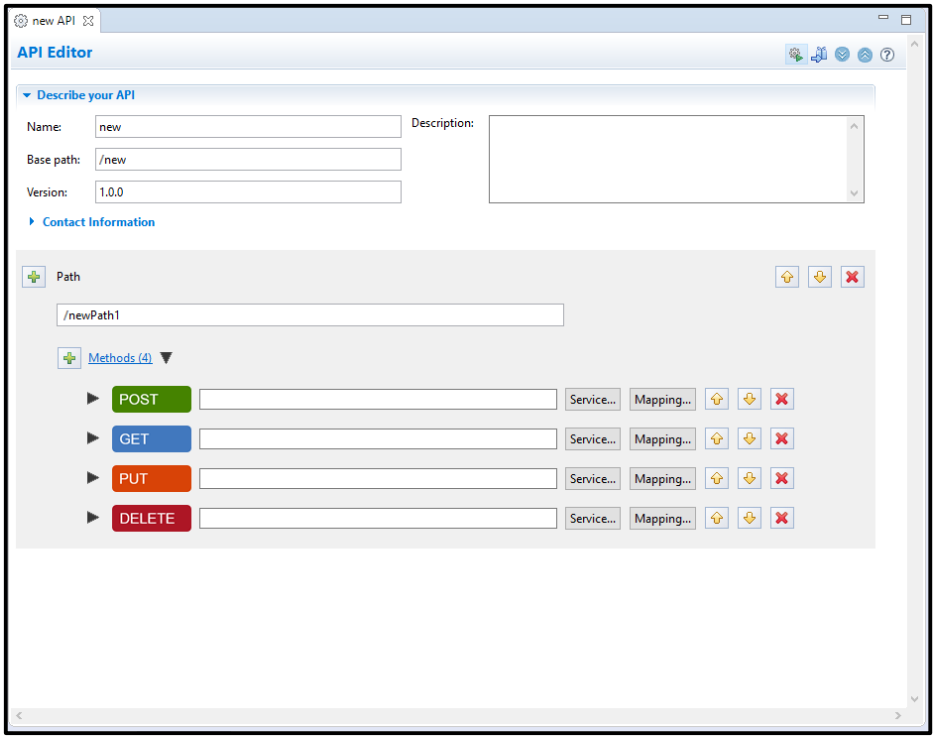
- **Open API Specification 2 (Support initially by z/OS Connect)**
  - Where the interactions with the z/OS resources were driven by the layout of the CICS COMMAREA or CONTAINER, the IMS or MQ messages or the Db2 REST service.
  - The z/OS resource interactions **determined** the contents of the API request and response messages and produced the specification document.
- **Open API Specification 3 (Supported by z/OS Connect as of the March 2022 service)**
  - As companies mature their API strategy, they begin to introduce API governance boards to drive consistency in their API design
  - As more public APIs are created, government and industry standards bodies begin to regulate and drive for standardization
  - This drives the need for “API first” functional mapping capabilities within the integration platform
  - The provided API design **determines** the contents of the API request and response messages provided by the specification documents. This document is consumed by z/OS Connect to describe the z/OS resource interactions

mitchj@us.ibm.com

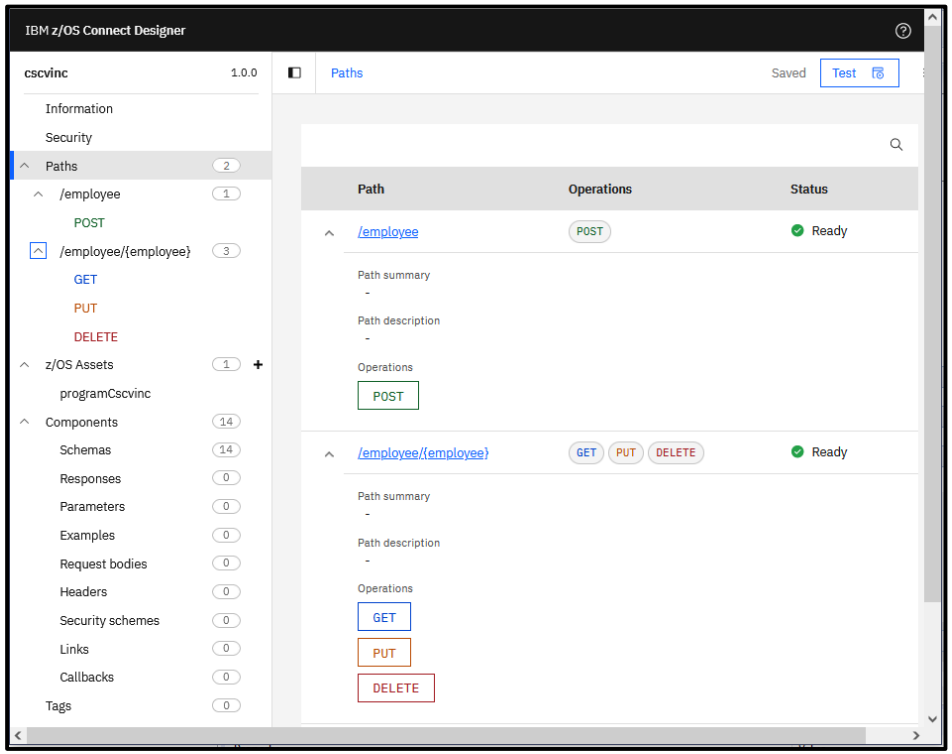
Both Open API2 and Open API3 are supported by z/OS Connect concurrently in different server instances*

The z/OS Connect OpenAPI 2 Palette versus the OpenAPI 3 API Designer

z/OS Connect API Toolkit (Eclipse)



z/OS Connect Designer (Designer Container)



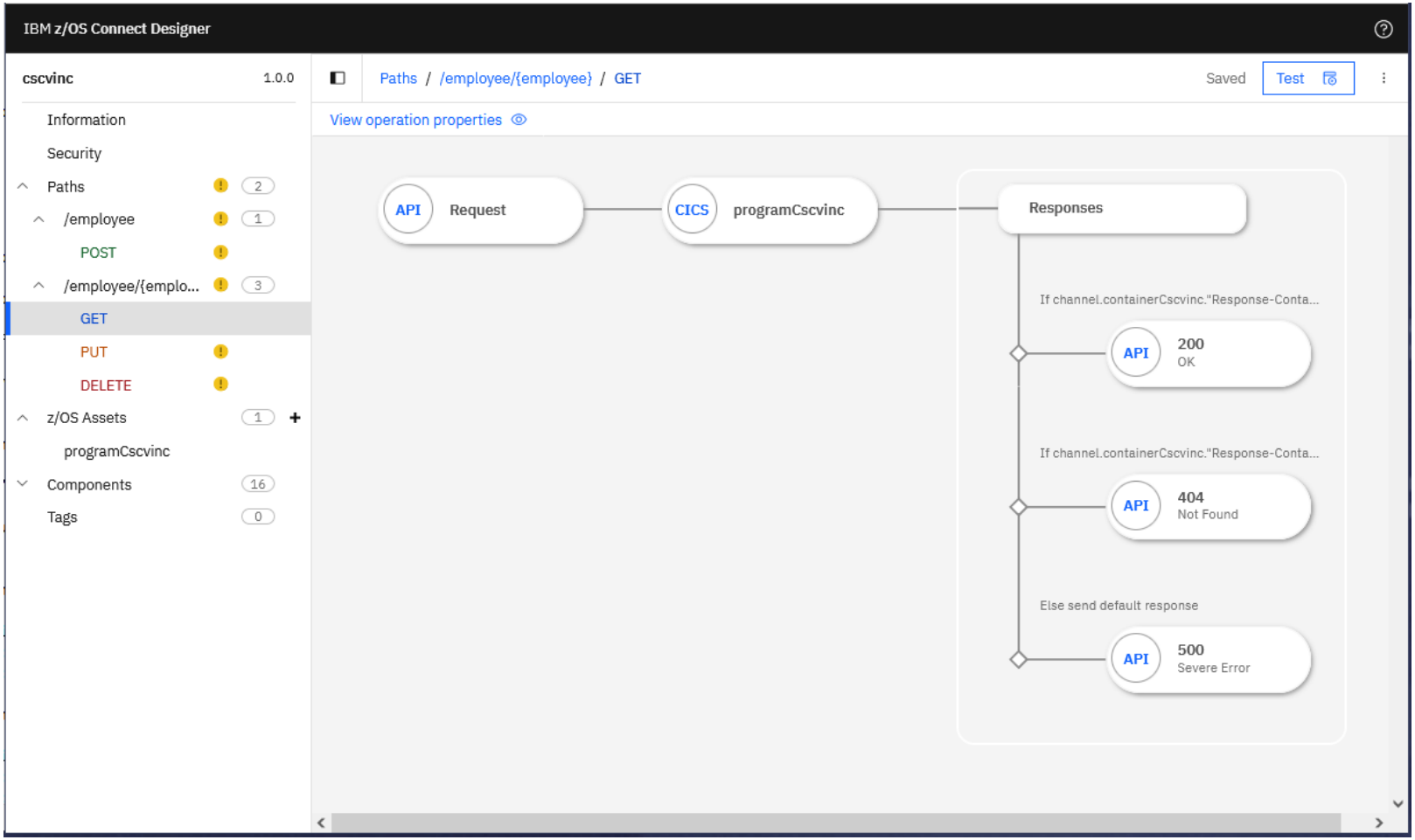
The API toolkit is used to **define the URI paths and methods.**

The API specification provides **predefined URI Paths and methods**.

mitchj@us.ibm.com

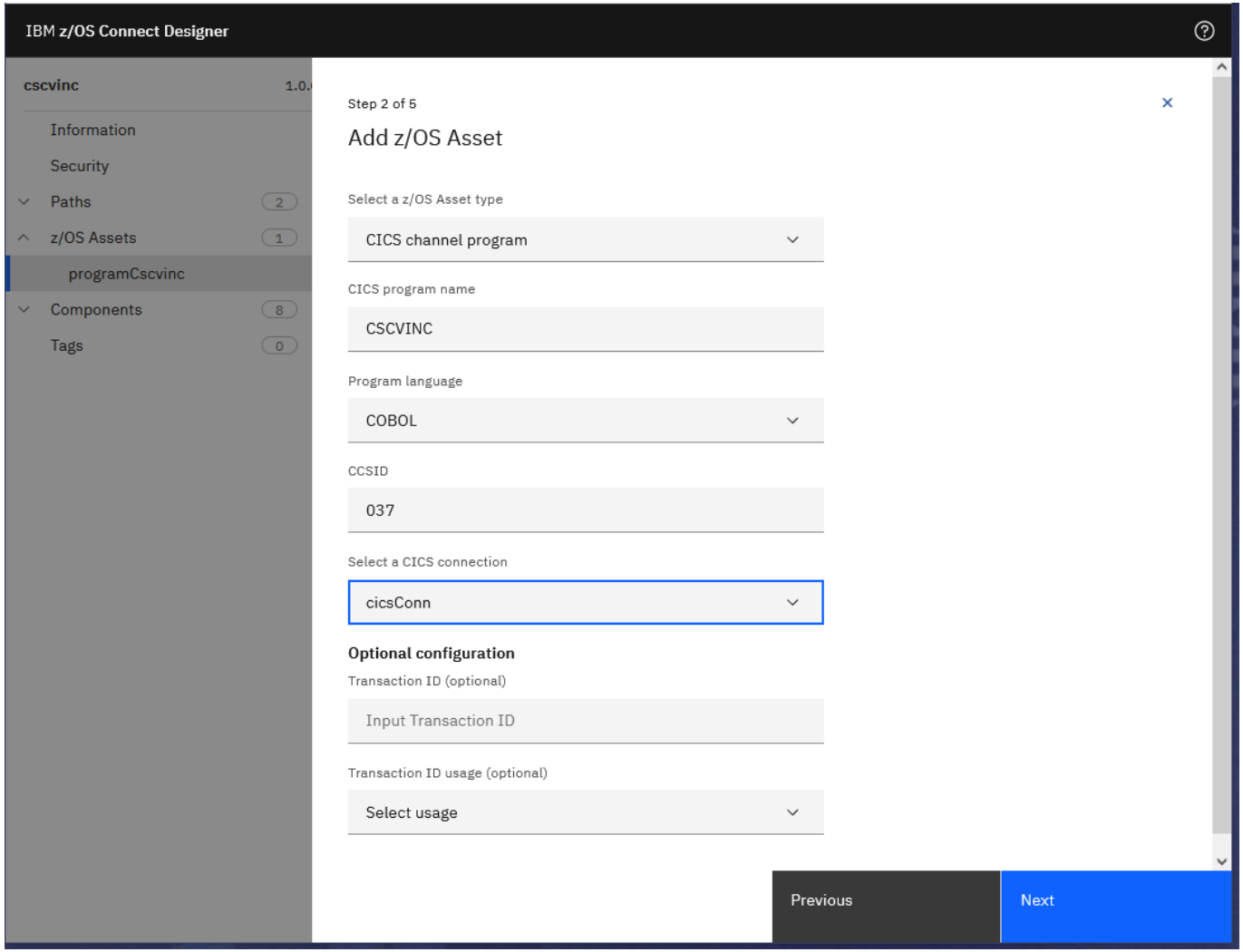
Developing an OpenAPI 3 API with the Designer

is started by importing the existing specification description of an API into the Designer



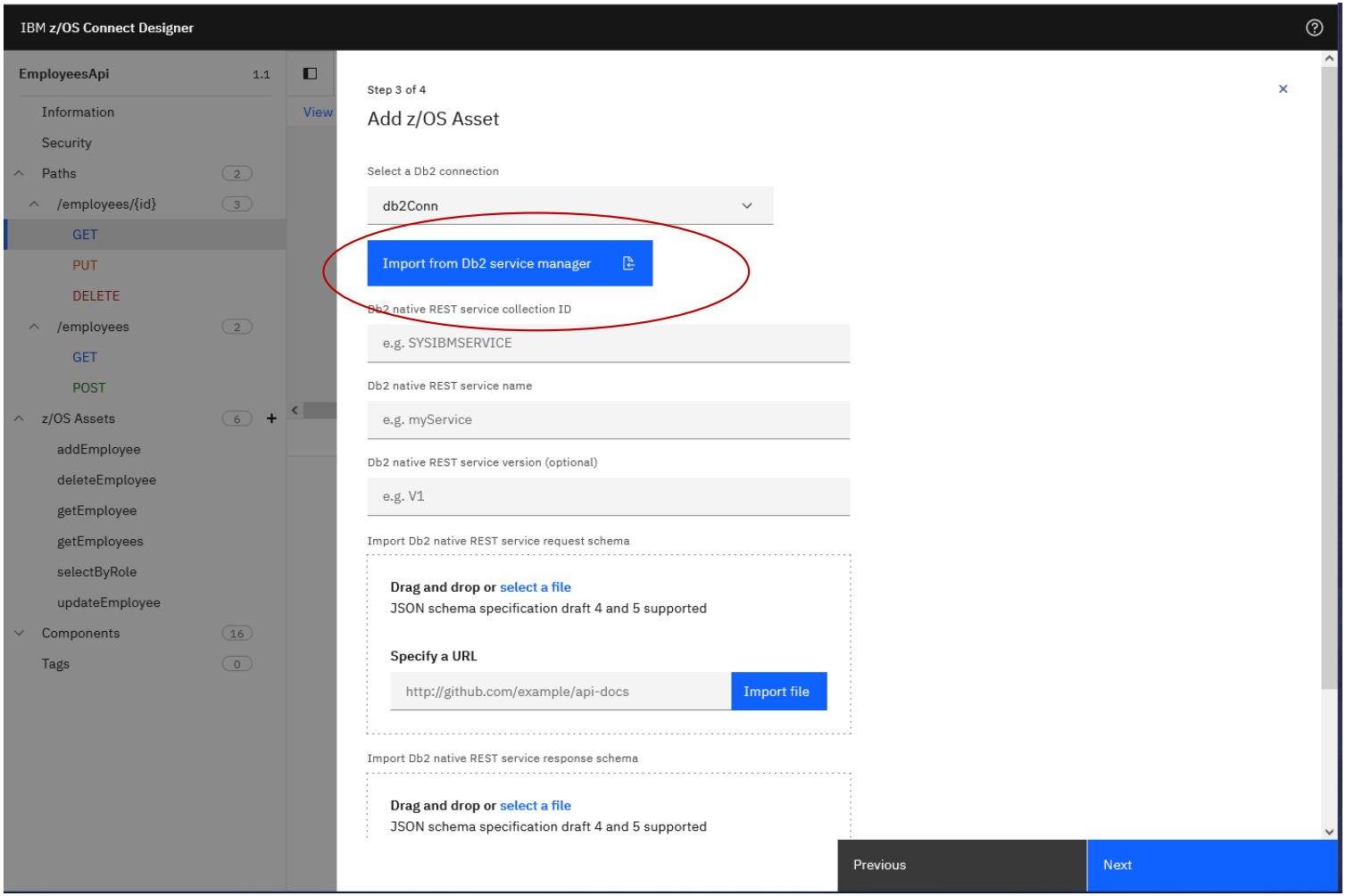
mitchj@us.ibm.com

Next by adding z/OS assets, e.g., for invoking a CICS program



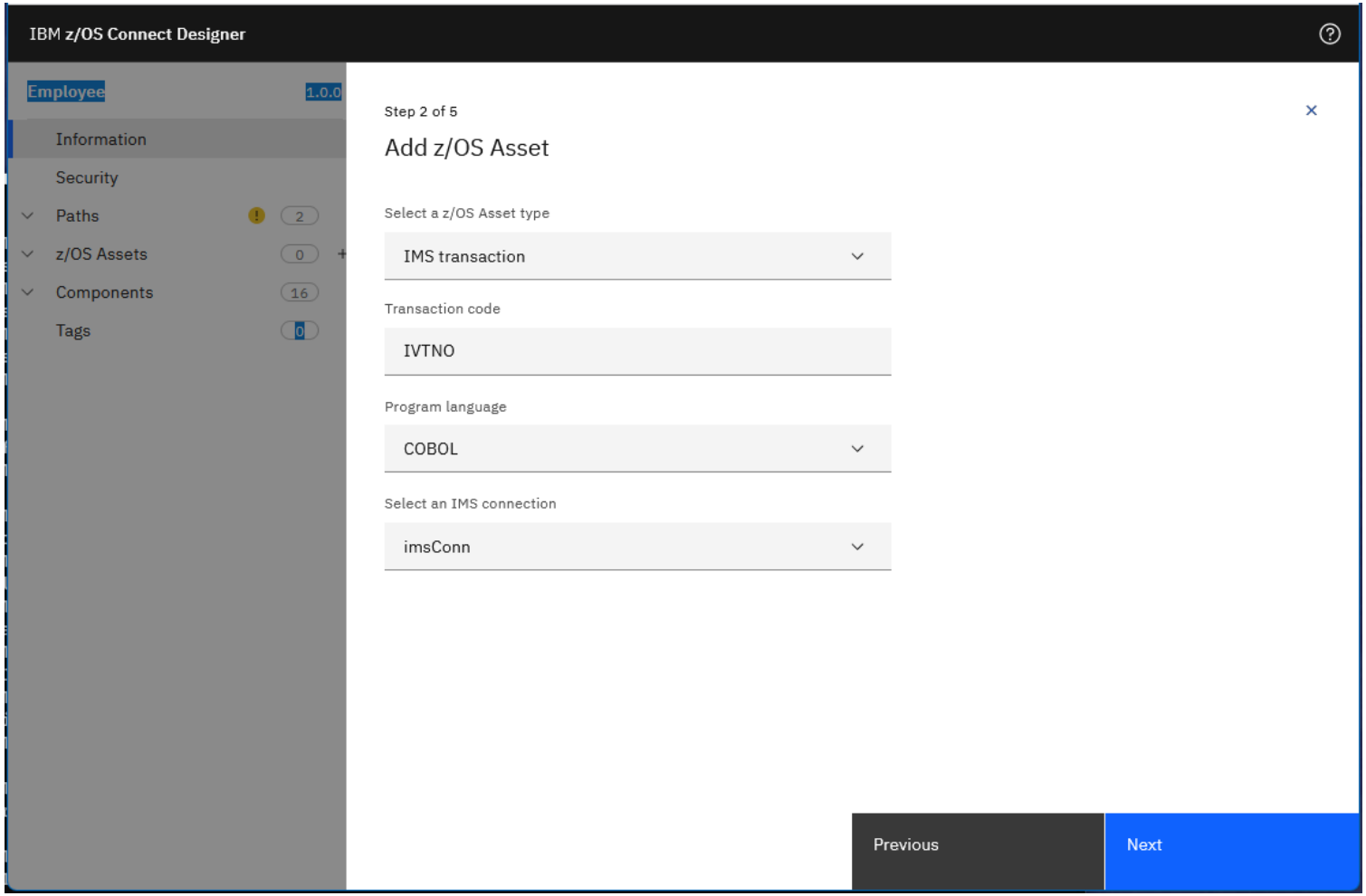
mitchj@us.ibm.com

Or for accessing a Db2 REST service



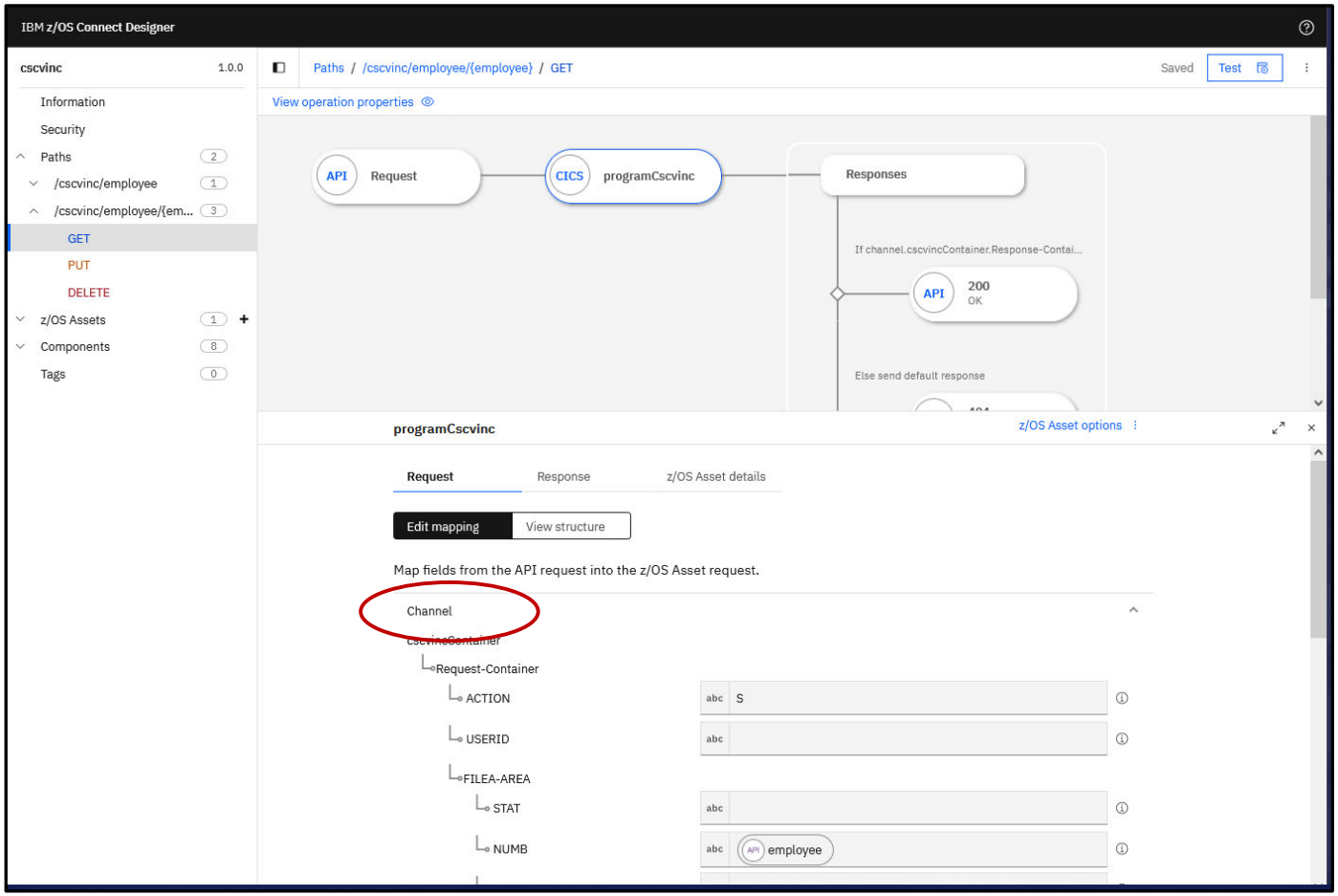
mitchj@us.ibm.com

Or for invoking an IMS transaction



mitchj@us.ibm.com

Next by mapping the API's *predefined* methods/request messages to the z/OS asset's "input"



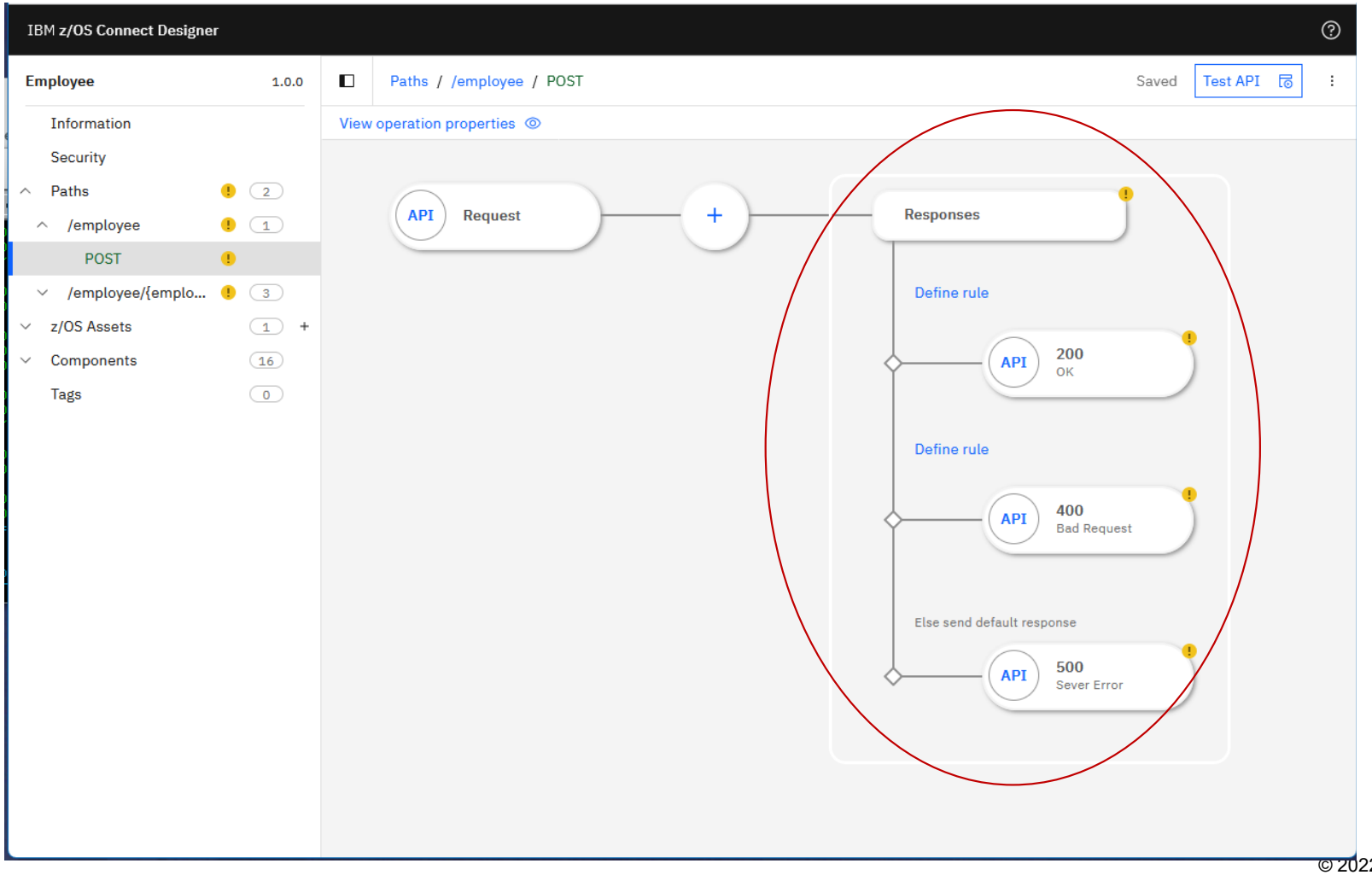
The z/OS asset's input can be a CICS COMMAREA or containers, am IMS Message or a Db2 REST service request message.

Map **(right to left)** the values provided by the API request message properties to the fields of the input "message" sent to the z/OS resources. And augment the z/OS request "message" as needed by the z/OS resource.

To: the z/OS "input" message ← From: the API request message body

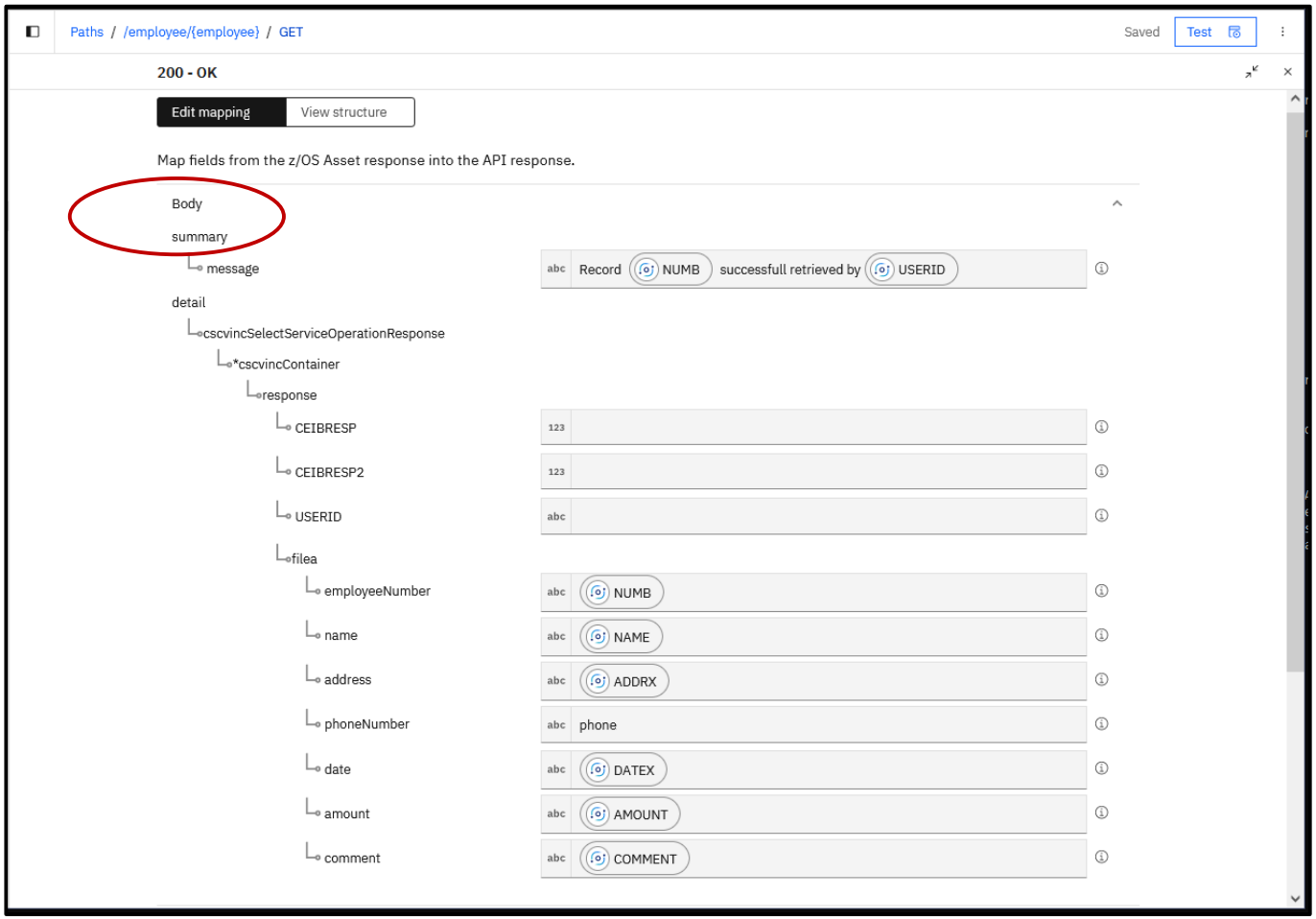
mitchj@us.ibm.com

Then mapping the API's HTTP predefined response codes with the asset's results



mitchj@us.ibm.com

Finally, by mapping the z/OS asset's "output" to the API's predefined response messages



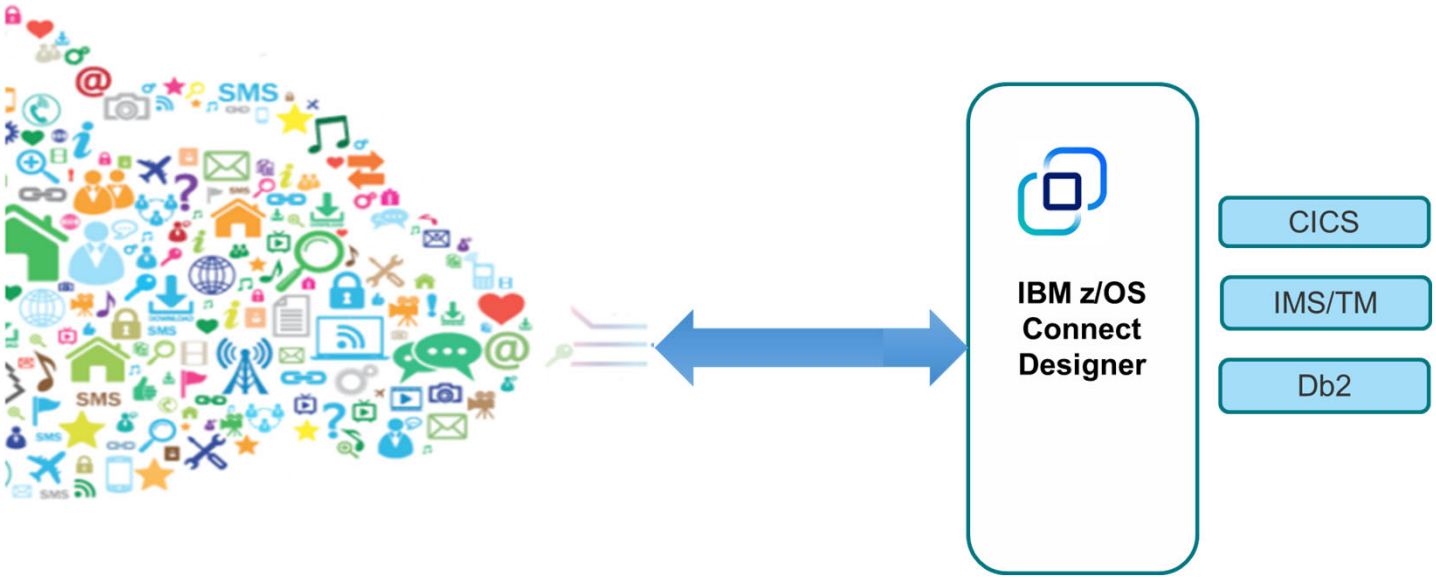
The z/OS asset's output can be a CICS COMMAREA or containers, am IMS Message or a Db2 REST service response message.

Map **(right to left)** the values returned by the z/OS asset's output to the corresponding API response properties and augment other API response message fields as necessary.

To: the API "response" message body ← From: the z/OS "output"

mitchj@us.ibm.com

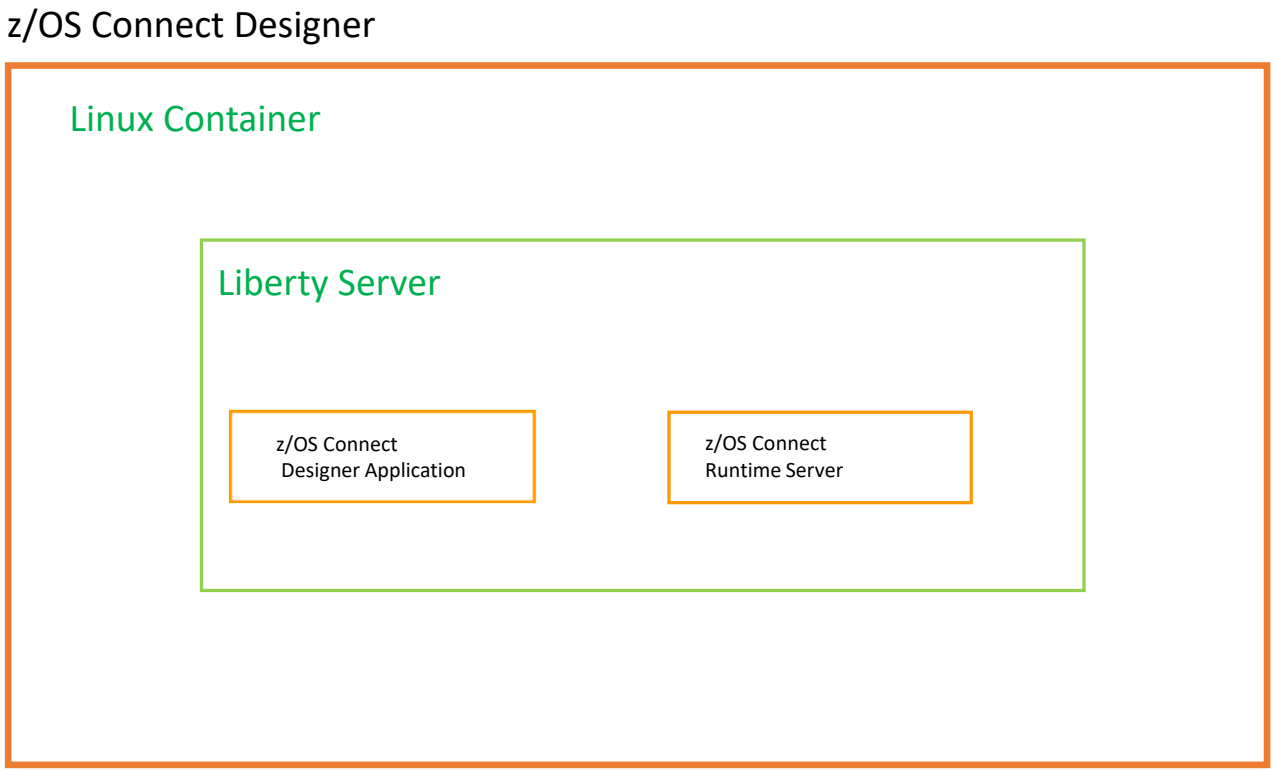
Deep Dive into using the z/OS Connect Designer to access z/OS resources



mitchj@us.ibm.com

Let’s begin by exploring the z/OS Connect Designer Topology

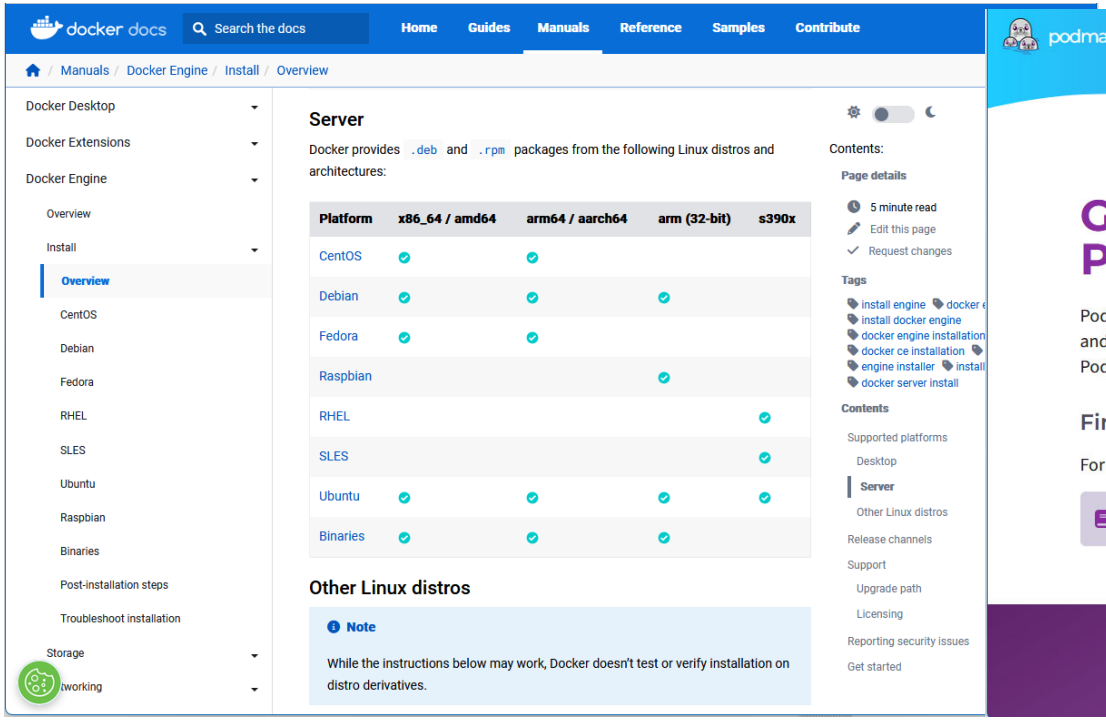
- A z/OS Connect Designer is composed of a self-contained Linux environment configured with a Liberty server running a z/OS Connect Designer application ***and*** a z/OS Connect runtime server, in total, known as a “container”.
- Running a Designer container requires a runtime environment.



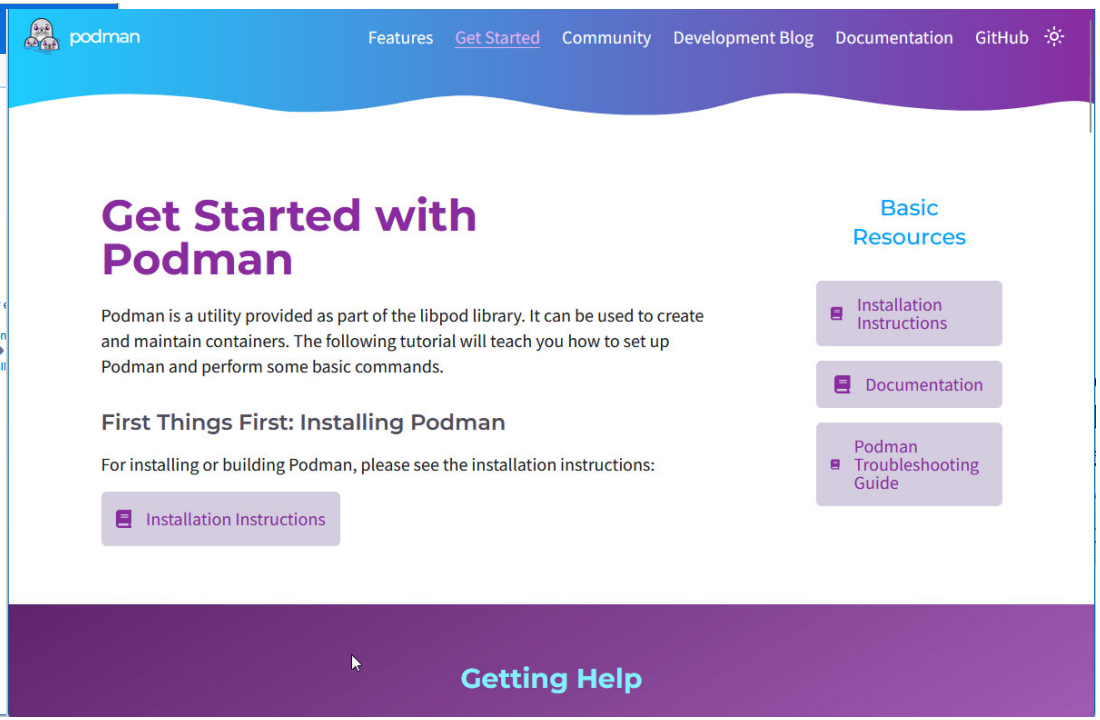
mitchj@us.ibm.com

Tech-Tip: Some suggested runtime environments that required no license

(one goal of this presentation is to show you why you don't need the "sexy" GUI tools)



https://docs.docker.com/engine/install/#server



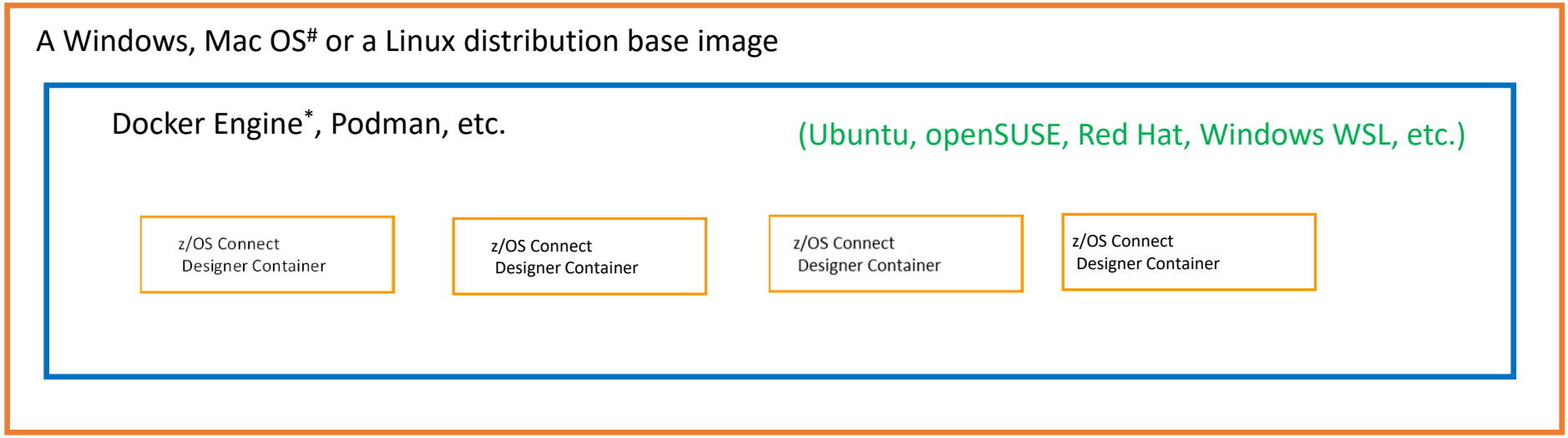
https://podman.io/get-started

Important: The command line interface (CLI) syntax is the same between the Docker Engine and Podman. Just change a Docker command from using the ***docker*** command to the ***podman*** command, e.g., ***docker ps -a*** when using Docker Engine becomes ***podman ps –a*** when using Podman.

mitchj@us.ibm.com

z/OS Connect Designer Runtime Options

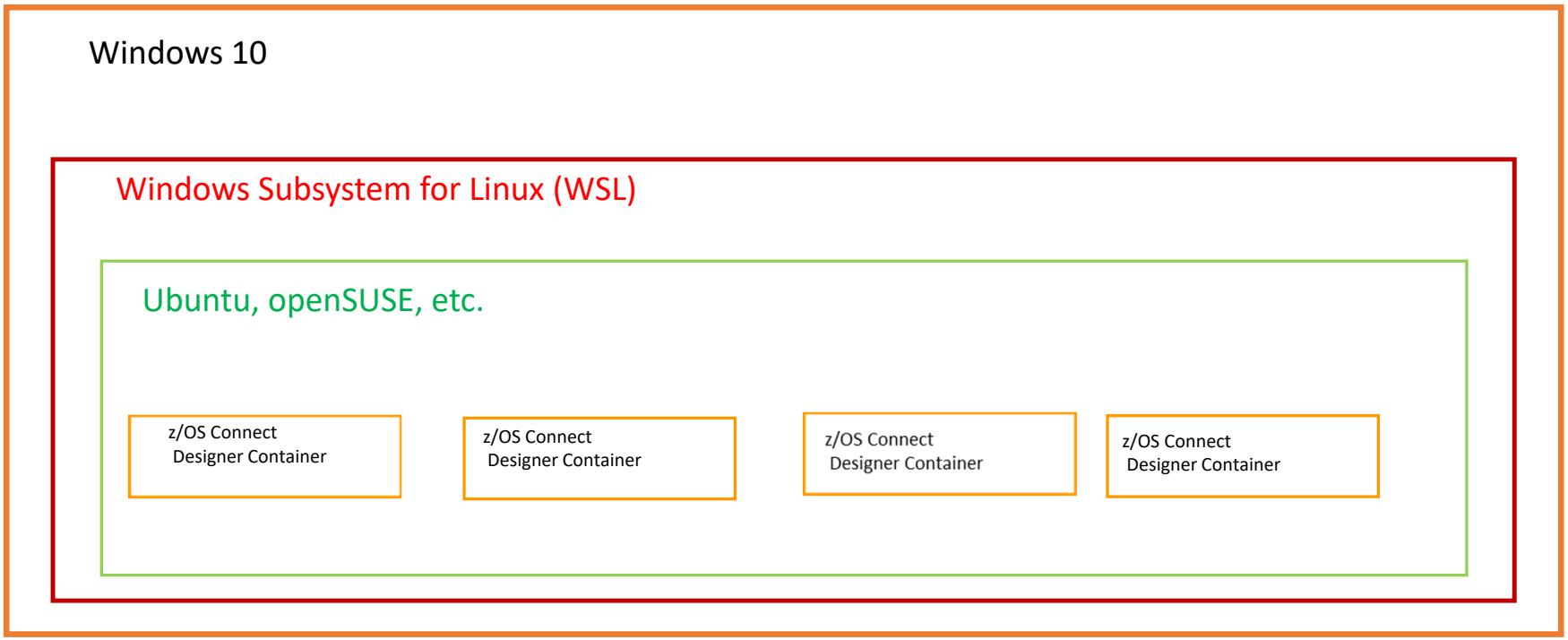
with one API per container, multiple Designer containers may be required



mitchj@us.ibm.com

#Warning, Machines with Mac M1 Pro processors may be problematic.
*Docker Engine is not an option on Macs, use Podman

Windows considerations

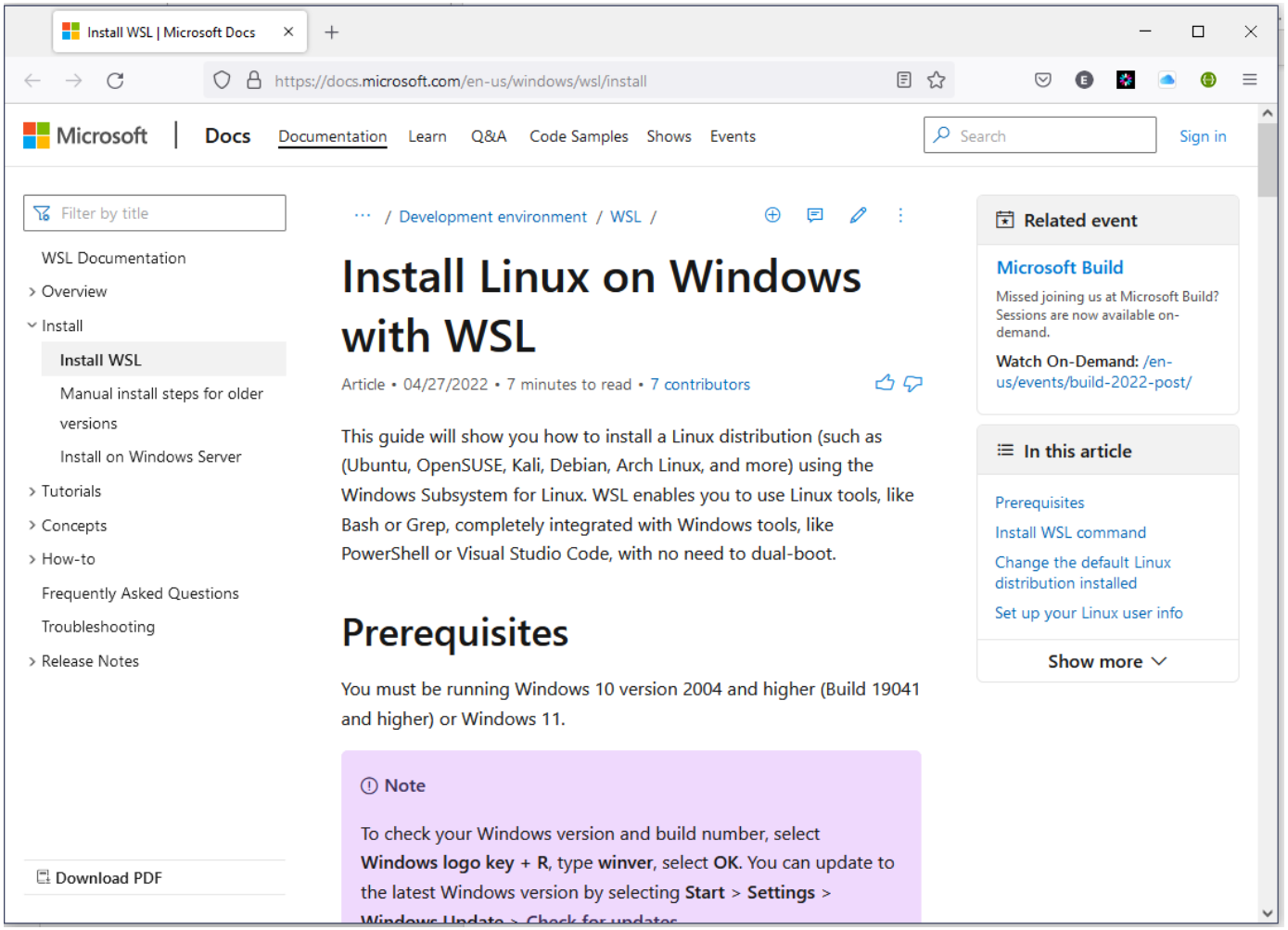


A z/OS Connect Designer is an application running in Liberty running in Linux (the container) which is running Linux (the container runtime) running in WSL running in Windows.

mitchj@us.ibm.com

Tech-Tip: Install Linux on Windows with WSL

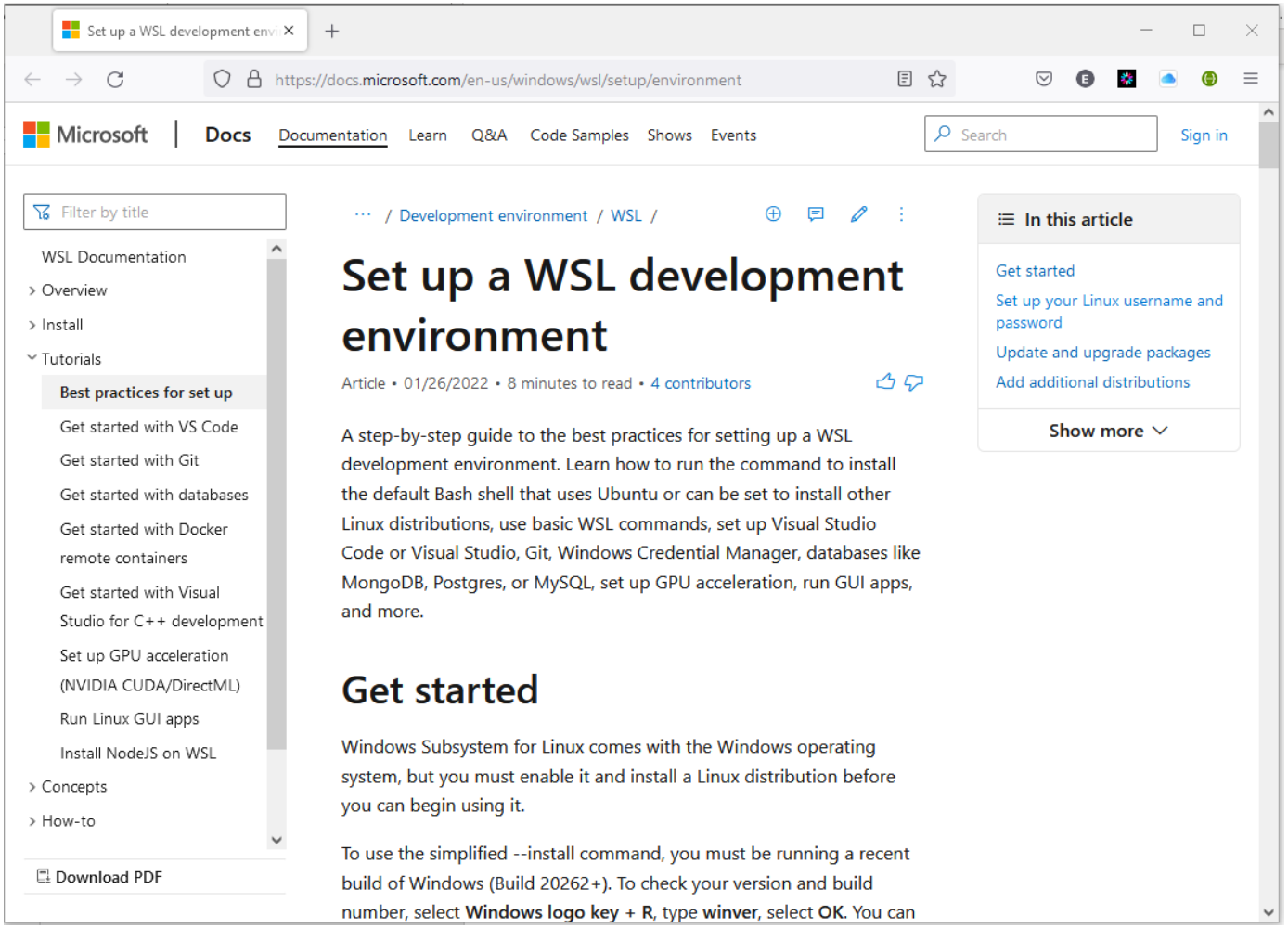
https://docs.microsoft.com/en-us/windows/wsl/install



mitchj@us.ibm.com

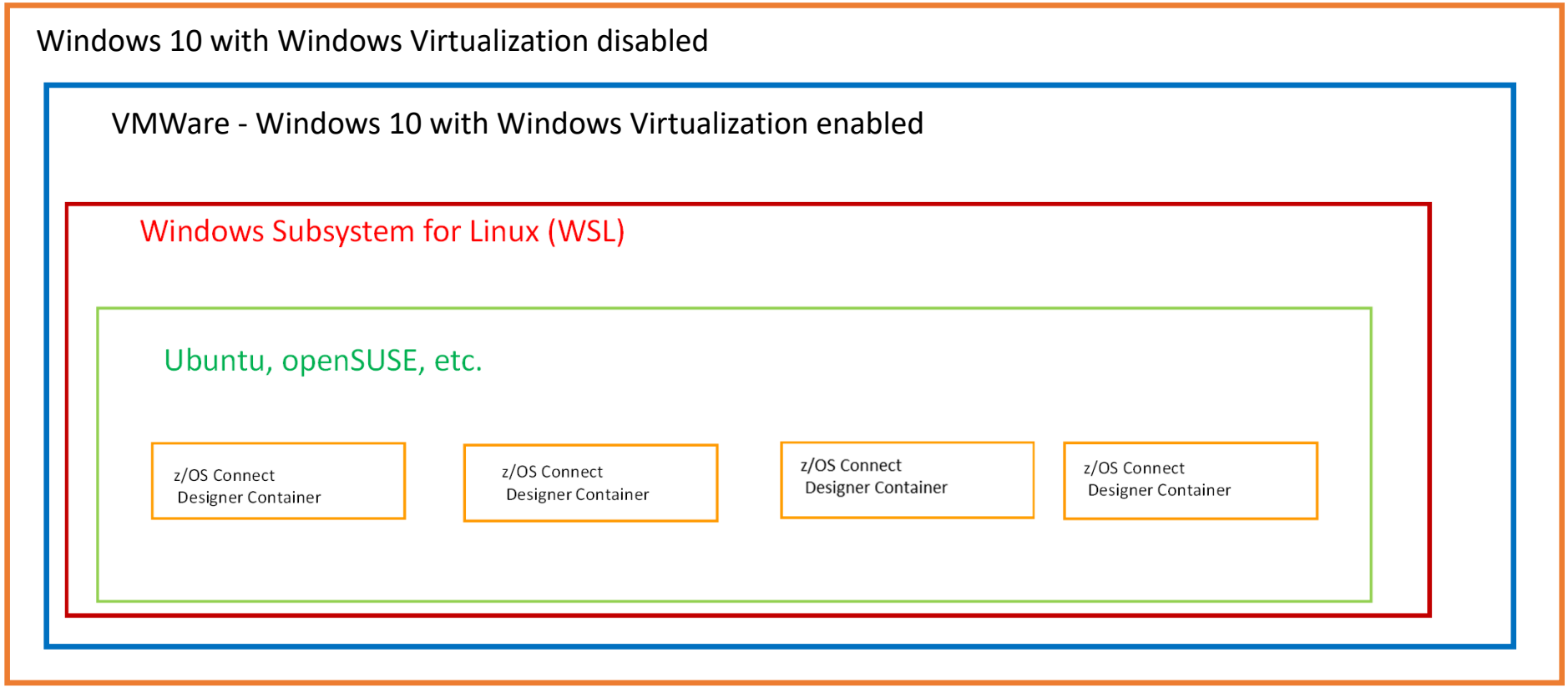
Tech-Tip: Set up a WSL development environment

https://docs.microsoft.com/en-us/windows/wsl/setup/environment



mitchj@us.ibm.com

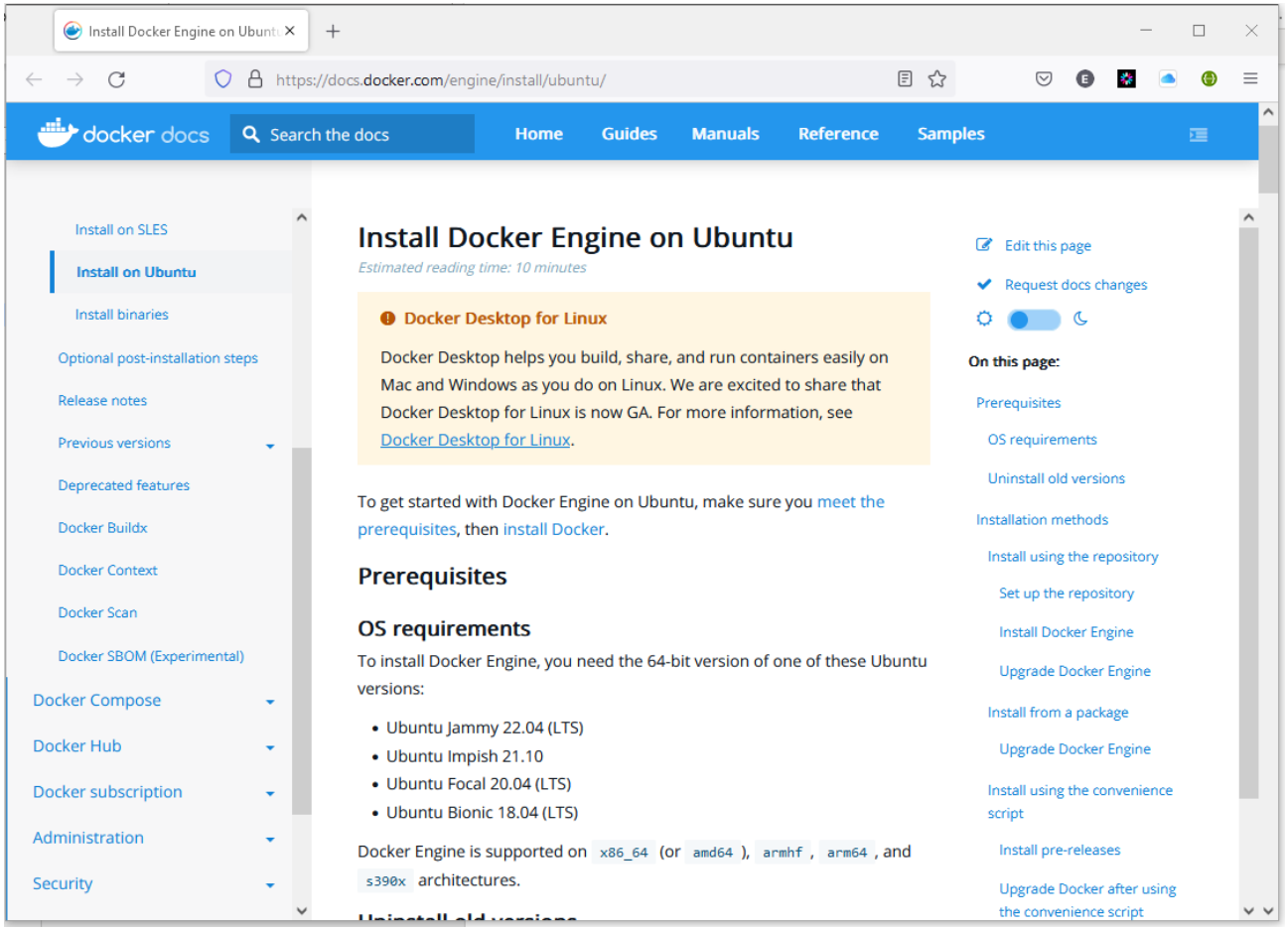
Tech Tip: z/OS Connect Designer Topology – VMWare Consideration



mitchj@us.ibm.com

Tech-Tip: Installing Docker Engine in a Linux distribution in WSL

https://docs.docker.com/engine/install/ubuntu/



mitchj@us.ibm.com

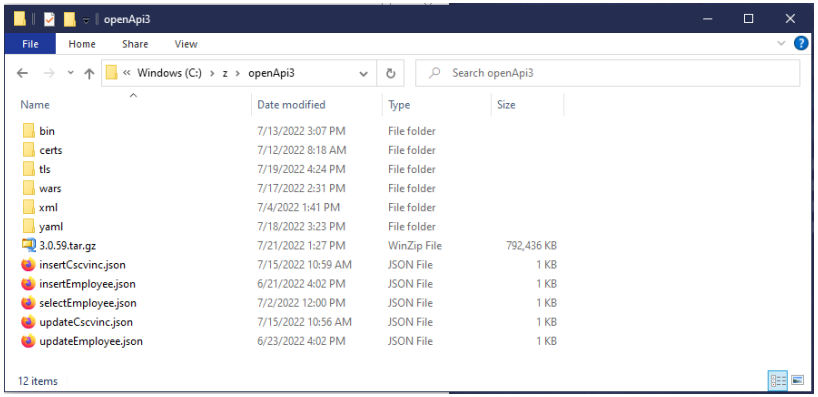
Using Linux commands as an alternative to using Docker Desktop

Useful Linux commands for accessing Windows files from the Linux runtime environment

Display Windows related filesystems

- *df | grep /mnt*

```
root:/home/workstation:> df | grep /mnt
tmpfs            1631068        0   1631068   0% /mnt/wsl
C:\            62271540 44278396  17993144  72% /mnt/c
root:/home/workstation:>
```



Copy files from the Windows host to the Linux container runtime (using the *sudo* command)

- *sudo cp /mnt/c/z/openApi3/xml/*.xml .*
- *sudo cp /mnt/c/z/openApi3/yaml/*.yaml .*

A dot means the current directory.

Or it is best to use the *su* command and just switch to root authority

- *su root*
- *cp /mnt/c/z/openApi3/xml/*.xml .*
- *cp /mnt/c/z/openApi3/yaml/*.yaml .*

A dot means the current directory.

Copy files from the Linux container runtime to the Windows host

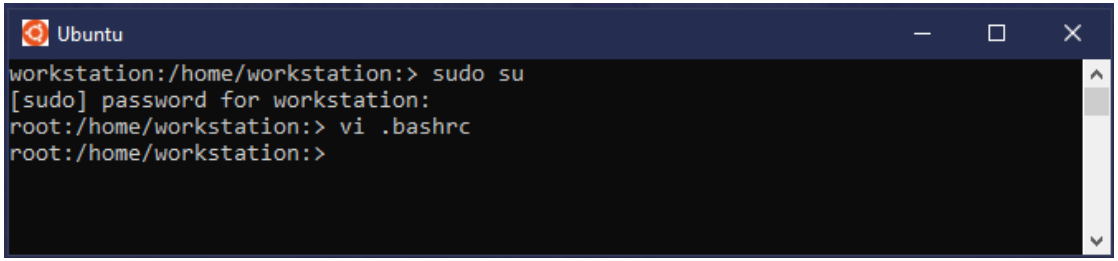
- *cp /home/workstation/docker/cscvinc/project/build/libs/api.war /mnt/c/z/openApi3/wars/cscvinc.war*

mitchj@us.ibm.com

Tech-Tip: Customized the Linux shell environment

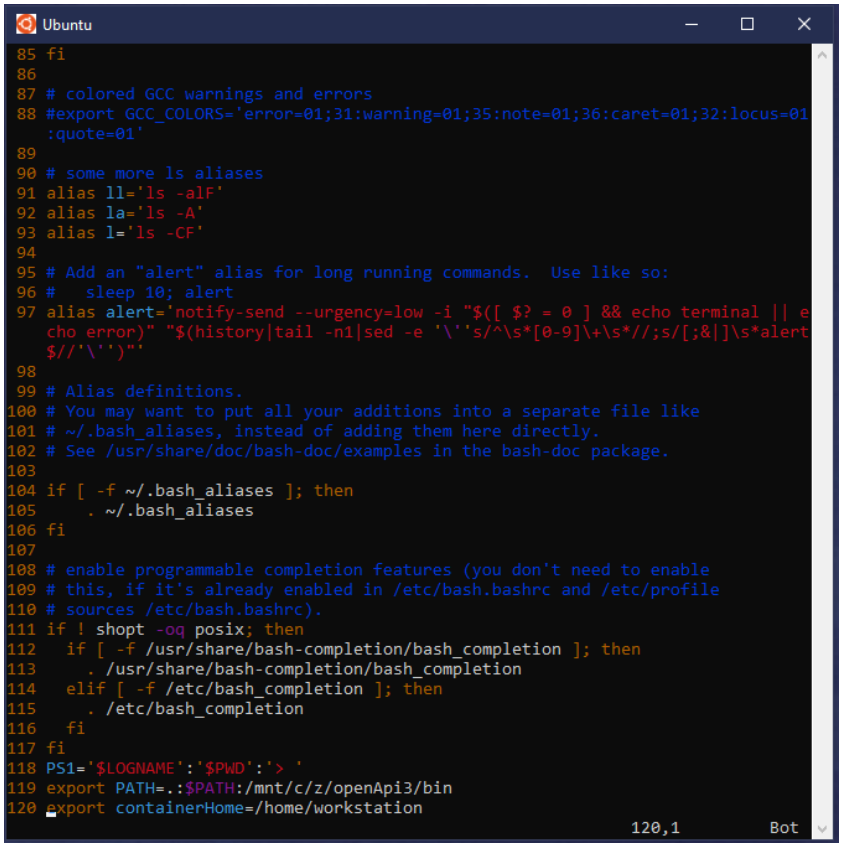
- Add these lines to file ***.bashrc*** in the Linux home directory

```
PS1='$LOGNAME':'$PWD':'> '
export PATH=.:$PATH:/mnt/c/z/openApi3/bin
export containerHome=/home/workstation
```



- Create a file named ***.exrc*** in the Linux home directory

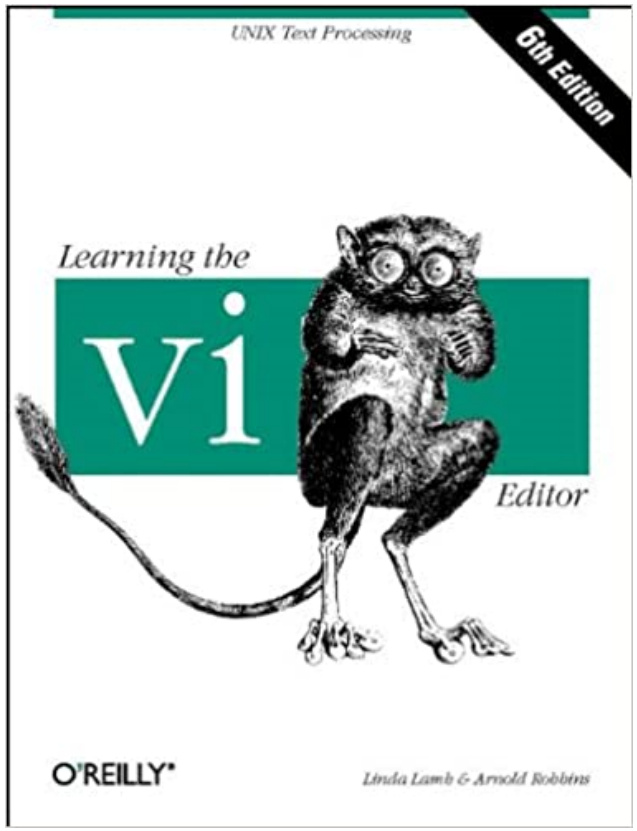
```
set showmode
set redraw
set wrapmargin=3
set nu
```



mitchj@us.ibm.com

Tech-Tip: Visual Editor (vi) Hints and Tips

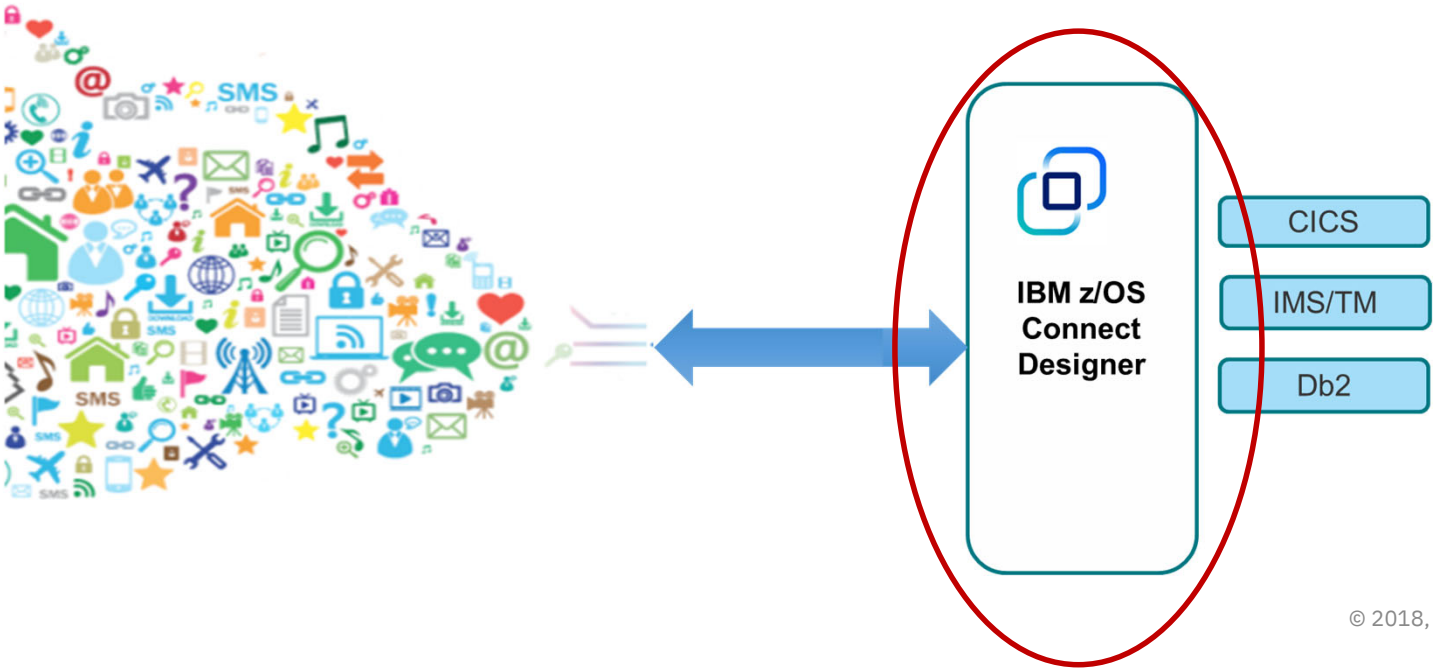
If you are going need to edit Linux files, I highly recommend this book for learning how to use the vi editor.



Learning the vi Editor (Nutshell Handbooks) Sixth Edition

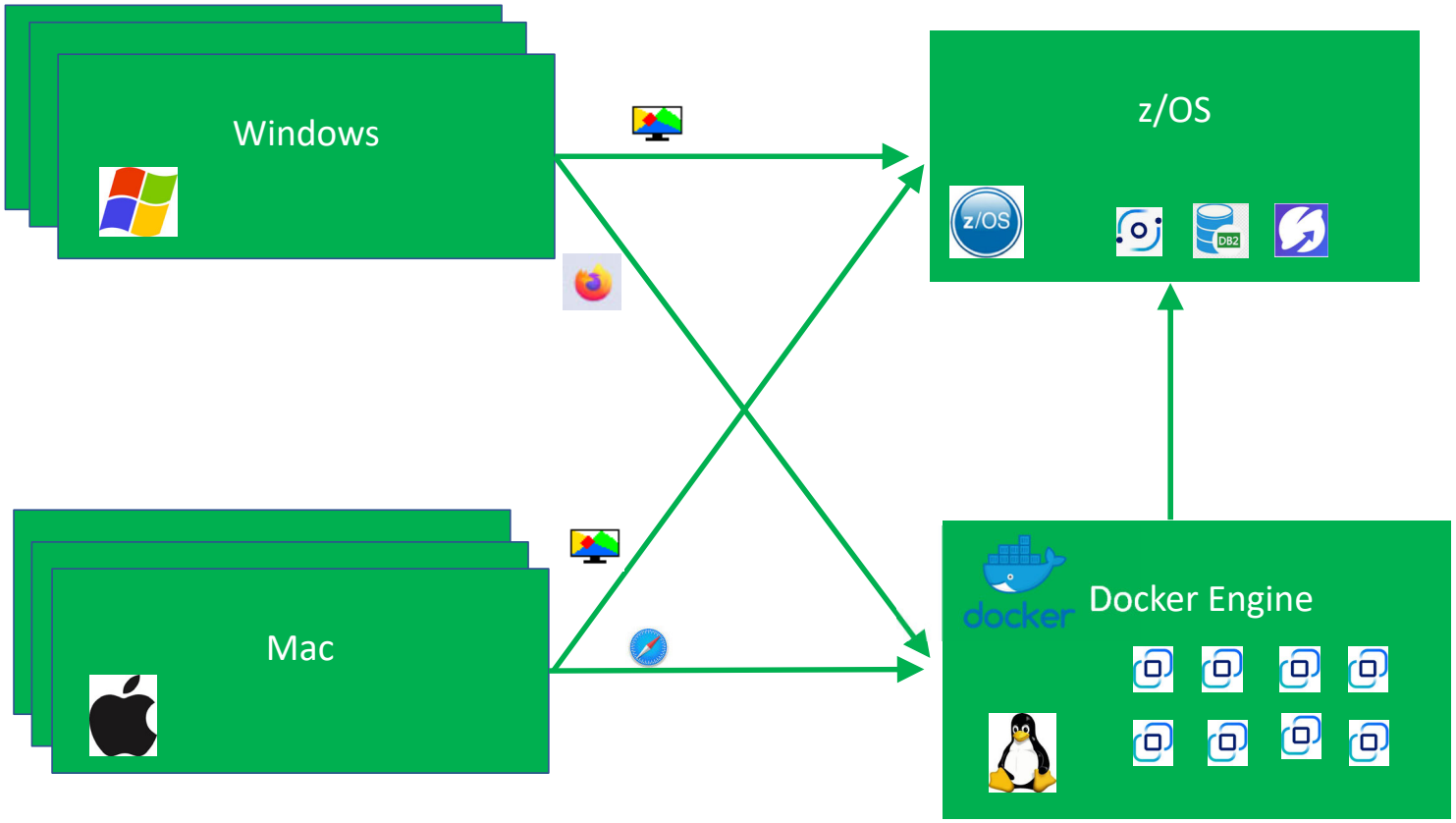
mitchj@us.ibm.com

Configuring and managing the Designer Container



mitchj@us.ibm.com

My z/OS Connect Designer Environment



mitchj@us.ibm.com

The WSC recommended default initial configuration file, e.g., *docker-compose.yaml*

```
version: "3.2"
services:
    zosConnect:
        image: icr.io/zosconnect/ibm-zcon-designer:3.0.68
        user: root
        environment:
                - BASE_PATH=basePath
                - CICS_USER=USER1
                - CICS_PASSWORD=USER1
                - CICS_HOST=wg31.washington.ibm.com
                - CICS_PORT=1491
                - DB2_USERNAME=USER1
                - DB2_PASSWORD=USER1
                - DB2_HOST=wg31.washington.ibm.com
                - DB2_PORT=2446
                - IMS_USER=USER1
                - IMS_PASSWORD=USER1
                - IMS_HOST=wg31.washington.ibm.com
                - IMS_PORT=4000
                - IMS_DATASTORE=IVP1
                - HTTP_PORT=9080
        ports:
            - "9443:9443"
            - "9080:9080"
        volumes:
            - ./CatalogManagerApi:/workspace/project
            - ./project:/workspace/project
            - ./logs/:/logs/
            - ./certs:/config/resources/security/:ro
            - ./certs:/output/resources/security/
```

The *environment* section is equivalent to a Liberty server's bootstrap.properties files

The contents of these directories are retained across the creation and destruction of containers. This means the contents of these directories are not lost if a container is removed, deleted or updated.
Mapping → ./Linux directory:/Container directory

The container name is the combination of the project name (e.g., current directory) || zosConnect || _1

mitchj@us.ibm.com

WSC's recommended sequence of steps for creating a new container

Enter these Linux commands while in the container directory

1. Make a new Linux directory for the container
   mkdir sandbox
2. Change location to the new directory
   cd sandbox
3. Make a "configuration" path directory
   mkdir -p project/src/main/liberty/config
4. Copy server XML configuration file from the base image (Windows) to the project's "configuration" directory
   cp /mnt/c/z/openApi3/xml/* project/src/main/liberty/config
5. Make the certs and logs subdirectories
   mkdir certs
   mkdir logs
6. Copy the base *docker-compose.yaml* file from the base image (Windows) into the current directory
   cp /mnt/c/z/openApi3/yaml/docker-compose.yaml .
7. Edit *docker-compose.yaml* file and make the ports unique
   vi docker-compose.yaml
8. Start the container
   docker -compose up –d
9. Copy a server's default XML override files from Windows into a container's directory*
   docker cp /mnt/c/z/openApi3/xml/. sandbox_zosConnect_1:/config/configDropins/overrides

Tech-Tip: Name the Linux directory the same as the intended context root and base path of the API.

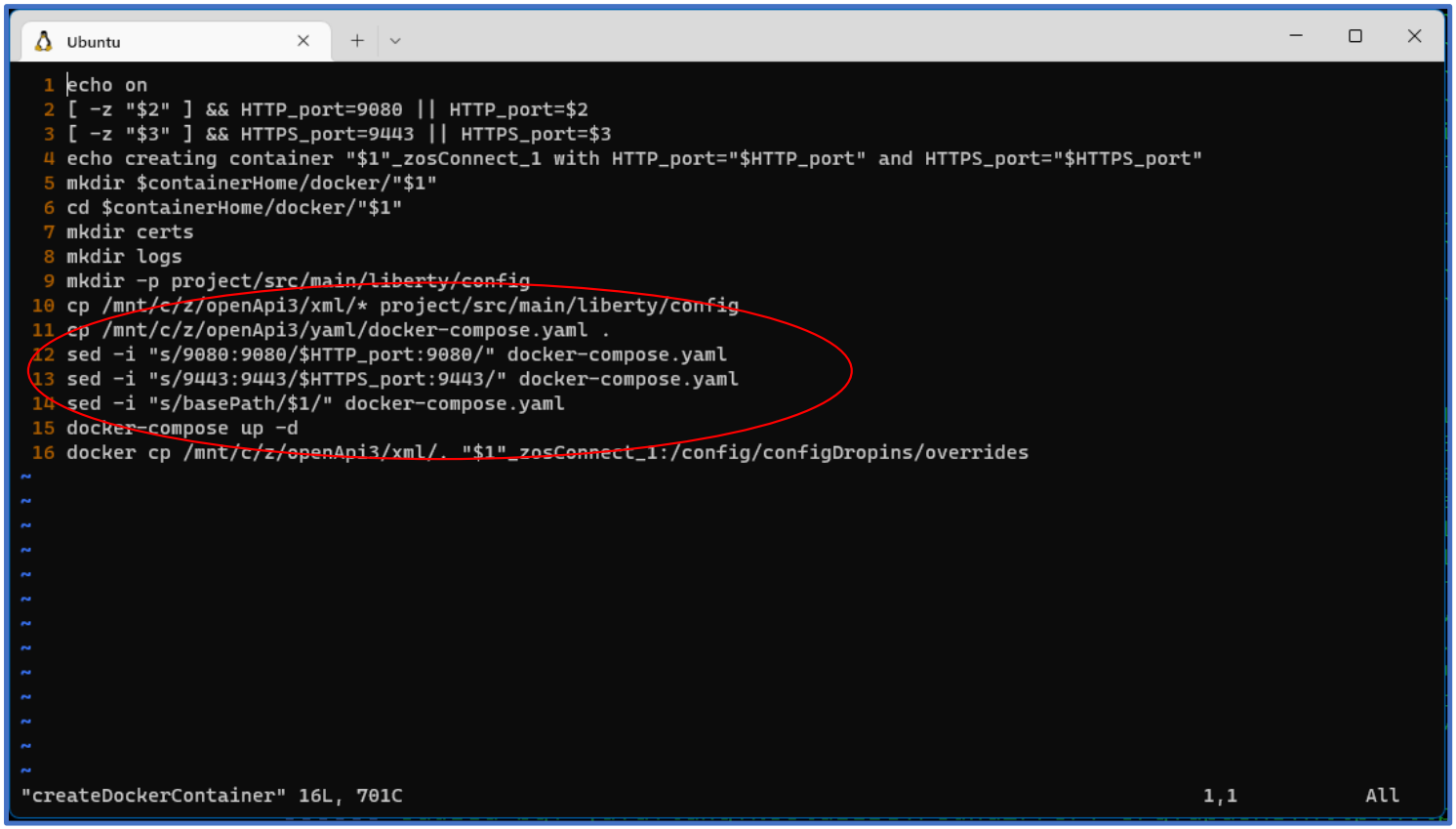
https://www.ibm.com/docs/en/zos-connect/zos-connect/3.0?topic=desktop-create-cics-tutorial-workspace-using-docker

mitchj@us.ibm.com

Steps 4 and 9 are redundant, but there were some initial timing issues regarding updating the Liberty server's configuration, step 9 addressed these issues

Tech-Tip: Create a Linux script named *createDockerContainer*

createDockerContainer containerName httpPort httpsPort



createDockerContainer sandbox 9090 9453

mitchj@us.ibm.com

Tech-Tip: the customized docker-compose.yaml file

```
version: "3.2"
services:
    zosConnect:
        image: icr.io/zosconnect/ibm-zcon-designer:3.0.68
        user: root
        environment:
               - BASE_PATH=sandbox
               - CICS_USER=USER1
               - CICS_PASSWORD=USER1
               - CICS_HOST=wg31.washington.ibm.com
               - CICS_PORT=1491
               - DB2_USERNAME=USER1
               - DB2_PASSWORD=USER1
               - DB2_HOST=wg31.washington.ibm.com
               - DB2_PORT=2446
                - IMS_USER=USER1
                - IMS_PASSWORD=USER1
                - IMS_HOST=wg31.washington.ibm.com
                - IMS_PORT=4000
                - IMS_DATASTORE=IVP1
               - HTTP_PORT=9080
        ports:
            - "9453:9443"
            - "9090:9080"
        volumes:
            - ./project:/workspace/project
            - ./logs/:/logs/
            - ./certs:/output/resources/security/
```

The container name will be **sandbox_zosConnect _1**

mitchj@us.ibm.com

Commands to refresh a container

- Remove the container (this does not remove or delete any existing application or configuration artifacts)

```
docker rm sandbox_zosConnect_1
```

- Set location to the container's Linux directory

```
cd /home/workstation/docker/sandbox
```

- Remove the subdirectories from under the project directory (this removes any application artifacts)

```
rm -r project/*
```

- Create the project directory subdirectory structure

```
mkdir -p project/src/main/liberty/config
```

- Copy the server XML files into the Linux config directory

```
cp /mnt/c/z/openApi3/xml/*  project/src/main/liberty/config
```

- Start the container

```
docker-compose up -d
```

- Copy a server's default XML files into the container's config overrides directory

```
docker cp /mnt/c/z/openApi3/xml/.
          sandbox_zosConnect_1:/config/configDropins/overrides
```

mitchj@us.ibm.com

Tech-Tip: Create a Linux script *refreshDockerContainer*

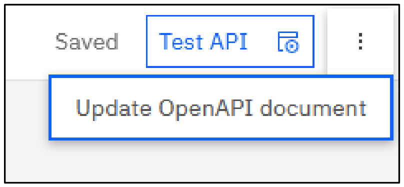
refreshDockerContainer containerName

```
echo refreshing container "$1"_zosConnect_1
docker stop "$1"_zosConnect_1
docker container rm "$1"_zosConnect_1
cd $containerHome/docker/"$1"
rm -r project/*
mkdir -p project/src/main/liberty/config
cp /mnt/c/z/openApi3/xml/* project/src/main/liberty/config
docker-compose up -d
docker cp /mnt/c/z/openApi3/xml/. "$1"_zosConnect_1:/config/configDropins/overrides
```

```
"refreshDockerContainer" 10L, 364C                9,1           All
```

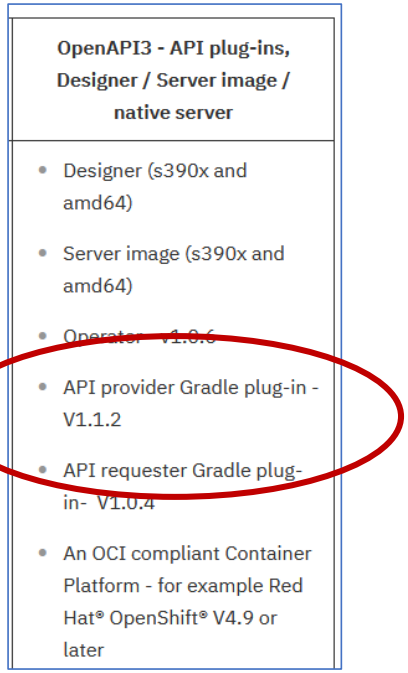
refreshDockerContainer sandbox

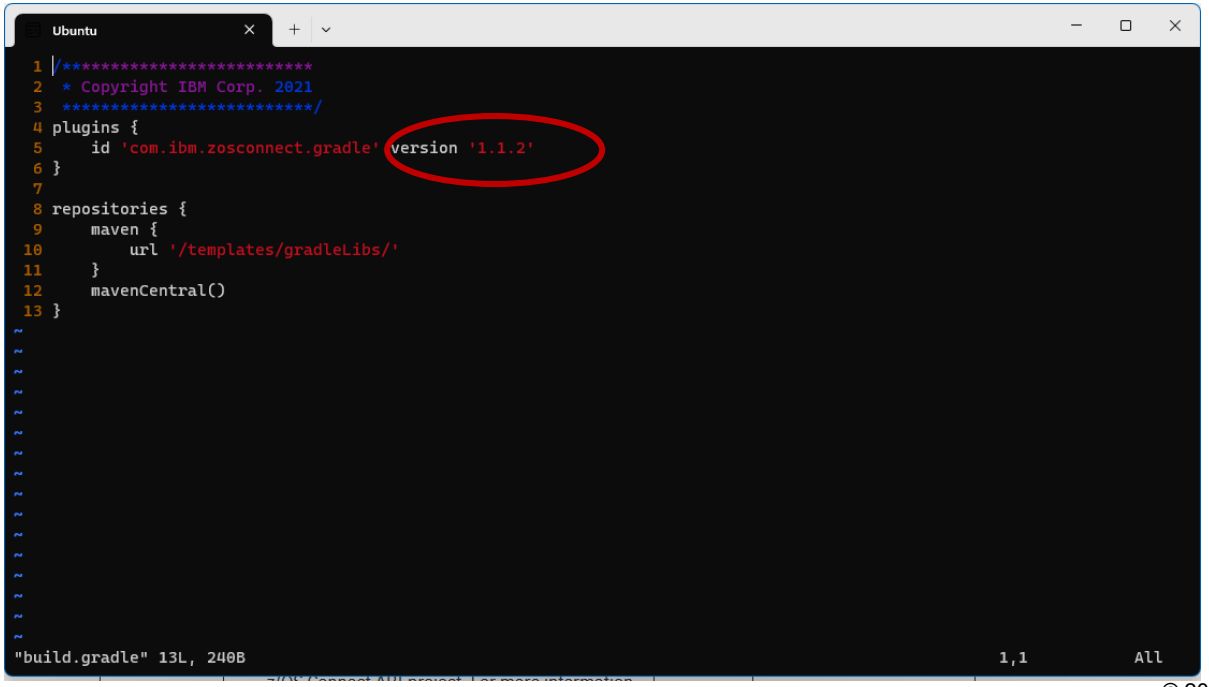
z/OS Connect service 3.0.71 added an enhancement to import an updated Open API3 document into an existing z/OS Connect API project,



Tech-Tip: Applying service to the Designer container

- Remove all contents from the *../certs* directory before restarting the container (the default keystore password has changed)
- Applying service can occasionally caused the *Test API* button to become disabled with message *BAQW0065E: API Project build failed. Review apiproject.log and messages.log for more information.* This occurs when the service has updated the Gradle plug-in version in the container. To know when the plug-in is updated, review the *What's New* information at URL [z/OS Connect - What's new in each release for OpenAPI2 and OpenAPI3] before applying service.
  - To resolve, update the *build.gradle* file in the *../project* directory so that the Gradle plug-in version in this file matches the updated container's version.





mitchj@us.ibm.com

The contents of my server XML configuration files

Contents of my /mnt/c/z/openApi3/xml

```
10 cp /mnt/c/z/openApi3/xml/* project/src/main/liberty/config
```

Ubuntu

```
mitchj:/home/mitchj:>ls -al /mnt/c/z/openapi3/xml
total 24
drwxrwxrwx 1 mitchj mitchj 4096 Oct 22 12:13 .
drwxrwxrwx 1 mitchj mitchj 4096 Sep 13 18:20 ..
-rwxrwxrwx 1 mitchj mitchj  191 Sep 21  2022 apiContext.xml
-rwxrwxrwx 1 mitchj mitchj 1556 Oct  9 10:53 basicSecurity.xml
-rwxrwxrwx 1 mitchj mitchj  452 Jul 14  2022 cics.xml
-rwxrwxrwx 1 mitchj mitchj  446 Jul  5  2022 db2.xml
-rwxrwxrwx 1 mitchj mitchj  512 Apr  5  2023 designerTrace.xml
-rwxrwxrwx 1 mitchj mitchj  662 Oct  9 10:53 features.xml
-rwxrwxrwx 1 mitchj mitchj  519 Apr  5  2023 ims.xml
-rwxrwxrwx 1 mitchj mitchj  376 Oct 22 12:13 keystore.xml
-rwxrwxrwx 1 mitchj mitchj  762 Sep 21  2022 webApplication.xml
mitchj:/home/mitchj:>
```

apiContext.xml | cics.xml

File Edit View

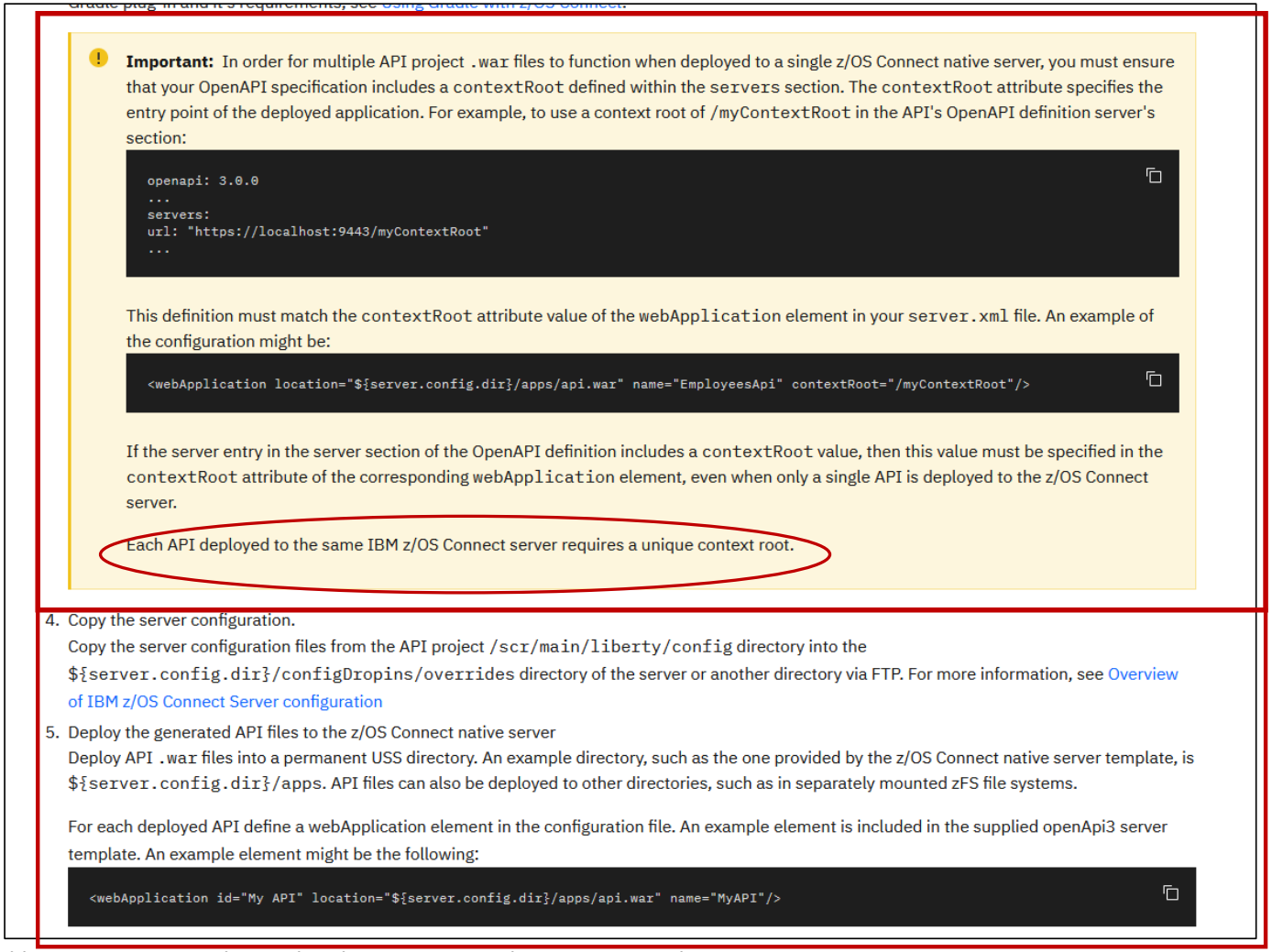
```
<?xml version="1.0" encoding="UTF-8"?>
<server description="IPIC connection to CICS">
    <featureManager>
        <feature>zosconnect:cics-1.0</feature>
    </featureManager>

    <zosconnect_cicsIpicConnection id="cicsConn"
        host="${CICS_HOST}"
        port="${CICS_PORT}"
        authDataRef="cicsCredentials" />

    <zosconnect_authData id="cicsCredentials"
        user="${CICS_USER}"
        password="${CICS_PASSWORD}" />

</server>
```

Ln 13, Col 29 | 100% | Unix (LF) | UTF-8

mitchj@us.ibm.com

A set common configuration XML files shared in all containers

Let’s stop for a moment: A lesson I learned the hard way. . . .
z/OS Native Server Considerations – API Provider Deployment



Important: In order for multiple API project `.war` files to function when deployed to a single z/OS Connect native server, you must ensure that your OpenAPI specification includes a `contextRoot` defined within the `servers` section. The `contextRoot` attribute specifies the entry point of the deployed application. For example, to use a context root of `/myContextRoot` in the API's OpenAPI definition server's section:

```
openapi: 3.0.0
...
servers:
url: "https://localhost:9443/myContextRoot"
...
```

This definition must match the `contextRoot` attribute value of the `webApplication` element in your `server.xml` file. An example of the configuration might be:

```
<webApplication location="${server.config.dir}/apps/api.war" name="EmployeesApi" contextRoot="/myContextRoot"/>
```

If the server entry in the server section of the OpenAPI definition includes a `contextRoot` value, then this value must be specified in the `contextRoot` attribute of the corresponding `webApplication` element, even when only a single API is deployed to the z/OS Connect server.

Each API deployed to the same IBM z/OS Connect server requires a unique context root.

4. Copy the server configuration.
   Copy the server configuration files from the API project `/scr/main/liberty/config` directory into the `${server.config.dir}/configDropins/overrides` directory of the server or another directory via FTP. For more information, see Overview of IBM z/OS Connect Server configuration
5. Deploy the generated API files to the z/OS Connect native server
   Deploy API `.war` files into a permanent USS directory. An example directory, such as the one provided by the z/OS Connect native server template, is `${server.config.dir}/apps`. API files can also be deployed to other directories, such as in separately mounted zFS file systems.

   For each deployed API define a webApplication element in the configuration file. An example element is included in the supplied openApi3 server template. An example element might be the following:

```
<webApplication id="My API" location="${server.config.dir}/apps/api.war" name="MyAPI"/>
```

https://www.ibm.com/docs/en/zos-connect/zos-connect/3.0?topic=overview-api-provider-devops

mitchj@us.ibm.com

z/OS Server Considerations – API Context Root

https://www.ibm.com/docs/en/zos-connect/zos-connect/3.0?topic=devops-zos-connect-server-overview

The drop-ins directory

The *drop-ins* directory, `/config/dropins` is a special directory that is supported by WebSphere® Application Server for Liberty. It allows `.war` files to be deployed and dynamically loaded into the running IBM z/OS Connect with no additional definitions that are required in the configuration file.

By default, z/OS Connect Designer deploys the API `.war` file to this directory. Using the same directory in your API container image simplifies the creation of that image because the configuration remains the same.

A directory other than drop-ins

This is required in any of the following situations:

- The API's OpenAPI definition server's section contains server entry that includes a context root value, which is not just /.
- Multiple APIs are to be deployed to the same IBM z/OS Connect container. Because the API `.war` file will be generated with a context root of /, and multiple API `.war` files in the same server must have unique context root values.

  You need to include a context root value (not /) in the API's OpenAPI definition server's section, for example to use a context root of `/MyCompany`:

```
openapi: 3.0.0
   ...
   servers:
      url: https://localhost:9443/MyCompany
   ...
```

- Requests to start an API require authentication only, without authorization, so the authorization roles need to be mapped to the WebSphere Application Server for Liberty special subject ALL_AUTHENTICATED_USERS. For more information, see How to define authorization roles.

If you choose not to use the drop-ins directory, you must alter the configuration that is used in z/OS Connect Designer during the creation of the API container image.

The samples as provided by z/OS Connect are not suitable as is for deployment to a z/OS Connect Native server.

- **Required to define applications and add context root**

```
<webApplication id="cics" contextRoot="/cics"  name="cicsAPI"
      location="${server.config.dir}apps/cscvinc.war"/>
<webApplication id="db2" contextRoot="/db2"  name="db2API"
      location="${server.config.dir}apps/employees.war"/>
```

mitchj@us.ibm.com

Contents of my /mnt/c/z/openApi3/xml/apiContext.xml file

- Default docker-compose YAML added a BASE_PATH environment variable

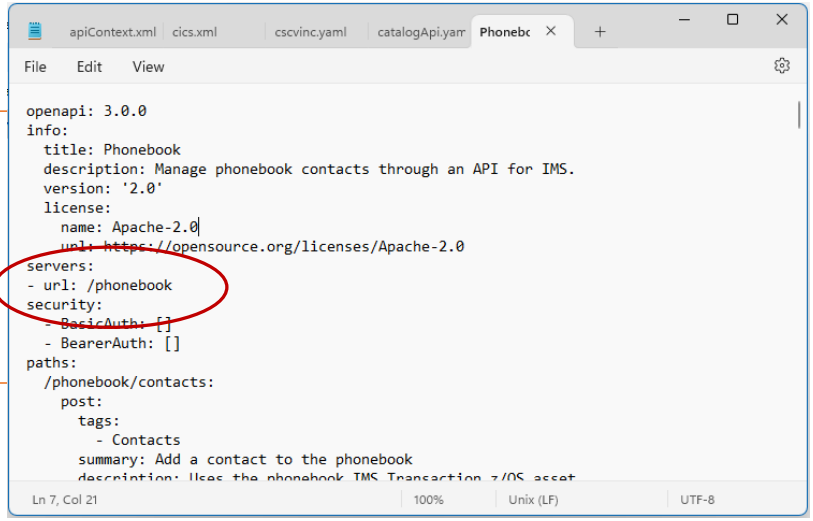
```
version: "3.2"
services:
    zosConnect:
        image: icr.io/zosconnect/ibm-zcon-designer:3.0.68
        user: root
        environment:
                - BASE_PATH=basePath
                - CICS_USER=USER1
                - CICS_PASSWORD=USER1
                - CICS_HOST=wg31.washington.ibm.com
```

- User script set the environment variable *basepath* is set to the name of the container

```
14 sed -i "s/basePath/$1/" docker-compose.yaml
```

- Environment variable BASE_PATH is used to set the name of the web application and context root in apiContext.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<server>

<webApplication id="myApi" name="${BASE_PATH}"
 contextRoot="/${BASE_PATH}"
 location="${server.config.dir}dropins/api.war" />

</server>
```



mitchj@us.ibm.com

Contents of my /mnt/c/z/openApi3/xml/cics.xml

- Default docker-compose YAML has these CICS related environment variables

```
- CICS_USER=USER1
- CICS_PASSWORD=USER1
- CICS_HOST=wg31.washington.ibm.com
- CICS_PORT=1491
```

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="IPIC connection to CICS">
    <featureManager>
        <feature>zosconnect:cics-1.0</feature>
    </featureManager>

    <zosconnect_cicsIpicConnection id="cicsConn"
        host="${CICS_HOST}"
        port="${CICS_PORT}"
        authDataRef="cicsCredentials" />

    <zosconnect_authData id="cicsCredentials"
        user="${CICS_USER}"
        password="${CICS_PASSWORD}" />

</server>
```

mitchj@us.ibm.com

Contents of my /mnt/c/z/openApi3/xml/db2.xml

- Default docker-compose YAML has these Db2 related environment variables

```
- DB2_USERNAME=USER1
- DB2_PASSWORD=USER1
- DB2_HOST=wg31.washington.ibm.com
- DB2_PORT=2446
- IMS_USER=USER1
```

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="Db2 zosconnect_db2Connection ">
    <featureManager>
        <feature>zosconnect:db2-1.0</feature>
    </featureManager>

<zosconnect_db2Connection id="db2Conn"
    host="${DB2_HOST}"
    port="${DB2_PORT}"
    credentialRef="commonCredentials" />

<zosconnect_credential id="commonCredentials"
    user="${DB2_USERNAME}"
    password="${DB2_PASSWORD}" />

</server>
```

mitchj@us.ibm.com

Contents of my /mnt/c/z/openApi3/xml/ims.xml

- Default docker-compose YAML has these IMS related environment variables

```
- IMS_USER=USER1
- IMS_PASSWORD=USER1
- IMS_HOST=wg31.washington.ibm.com
- IMS_PORT=4000
- IMS_DATASTORE=IVP1
```

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="zosconnect_imsConnection ">
<zosconnect_imsConnection id="imsConn"
        connectionFactoryRef="imsConnectionFactory"
        imsDatastoreName="${IMS_DATASTORE}"/>

    <connectionFactory id="imsConnectionFactory"
containerAuthDataRef="IMSCredentials">
        <properties.gmoa hostName="${IMS_HOST}" portNumber="${IMS_PORT}" />
    </connectionFactory>

    <authData id="IMSCredentials" user="${IMS_USER}" password="${IMS_PASSWORD}" />
</server>
```

mitchj@us.ibm.com

Contents of my /mnt/c/z/openApi3/xml/keystore.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <keyStore id="defaultKeyStore" password="Liberty"/>

</server>
```

Contents of my /mnt/c/z/openApi3/xml/features.xml

```
<server description="basic features">
  <!-- Enable features -->
      <featureManager>
            <feature>adminCenter-1.0</feature>
            <feature>restConnector-2.0</feature>
      </featureManager>
</server>
```

mitchj@us.ibm.com

Let’s stop again: z/OS Server Considerations – Adding Security Roles

https://www.ibm.com/docs/en/zos-connect/zos-connect/3.0?topic=authorization-how-define-roles

Procedure

1. Locate and open the OpenAPI document.
   If the OpenAPI document isn't imported into the Designer UI, then this is your original OpenAPI document.
   If the OpenAPI document is imported into the Designer UI, then this is the `openapi.yaml` or `openapi.json` file in the API project `src/main/api` directory. This might be in your local Designer workspace or might be stored in a Source Control Manager.
   Open the OpenAPI document in edit mode.

2. Optional: Define the roles that apply to all operations in the API.
   Define the `x-ibm-zcon-roles-allowed` in the root of the OpenAPI definition, where the value is an array of role names.

This seems to be recommending changing the API after it has been tested and/or deployed.

mitchj@us.ibm.com

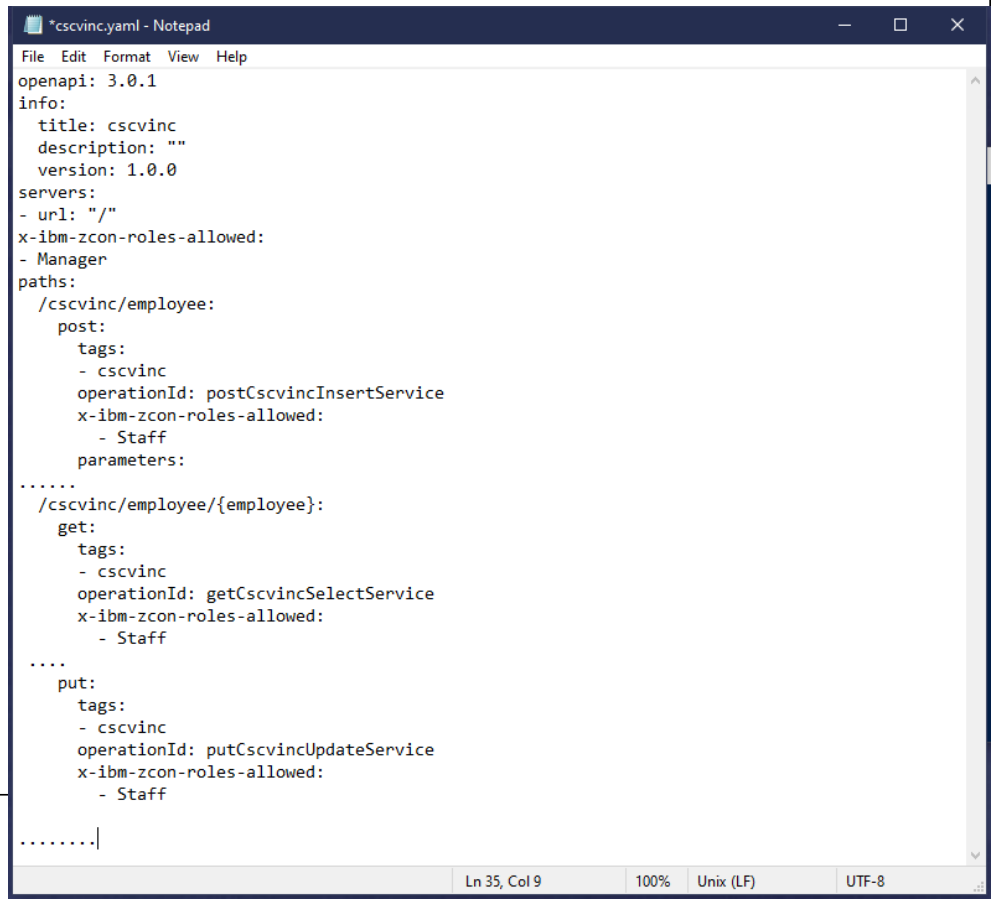
Contents of my /mnt/c/z/openApi3/xml/basicSecurity.xml (1 of 2)

```
<server description="basic security">

    <!-- Enable features -->
    <featureManager>
        <feature>appSecurity-2.0</feature>
    </featureManager>

    <webAppSecurity allowFailOverToBasicAuth="true" />

    <basicRegistry id="basic" realm="zosConnect">
        <user name="Fred" password="fredpwd" />
        <user name="user1" password="user1" />
        <user name="user2" password="user2" />
        <user name="user3" password="user3" />
        <group name="Manager">
           <member name="Fred"/>
        </group>
        <group name="Staff">
           <member name="Fred"/>
           <member name="user1"/>
           <member name="user2"/>
        </group>
     </basicRegistry>
```

mitchj@us.ibm.com

Contents of my /mnt/c/z/openApi3/xml/basicSecurity.xml (2 of 2)

```
<administrator-role>
        <user>Fred</user>
        <group>staffGroup</group>
    </administrator-role>

<authorization-roles id="Manager">
        <security-role name="Manager">
            <group name="managerGroup"/>
        </security-role>
     </authorization-roles>
     <authorization-roles id="Staff">
        <security-role name="Staff">
            <group name="staffGroup"/>
        </security-role>
     </authorization-roles>

</server>
```



mitchj@us.ibm.com

Tech-Tip: Rember these roles related directly to z/OS EJB roles for defined to SAF resources

```
<safCredentials unauthenticatedUser="WSGUEST" profilePrefix="BBGZDFLT" />

<webApplication id="catalogManager" name="catalogManager"
              location="${server.config.dir}/apps/api.war" contextRoot="/catalogManager" />

<safRoleMapper profilePattern=%profilePrefix%.%resourceName%.%role%
```

```
openapi: 3.0.0
. . .
servers:
- url: /
x-ibm-zcon-roles-allowed:
- Manager
. . .
paths:
  /items:
    get:
     operationId: itemsGet
     . . .
  /items/{id}:
    get:
     . . .
     operationId: itemsIdGet
     x-ibm-zcon-roles-allowed:
        - Staff
 /orders:
     post:
     . . .
     operationId: ordersPost
     x-ibm-zcon-roles-allowed:
       - Staff
```

From the OpenApi document, the value for %role% would be either ***Manager*** *or* ***Staff****.*

So, the required SAF EJB roles to be defined would be:

- *BBGZDFLT.catalogManager.**Man***

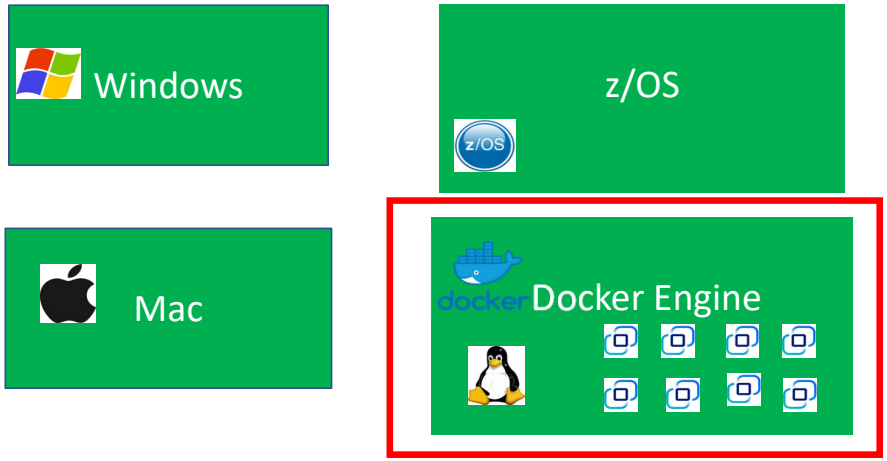
- ***ager***
- *BBGZDFLT.catalogManager.**Staff***

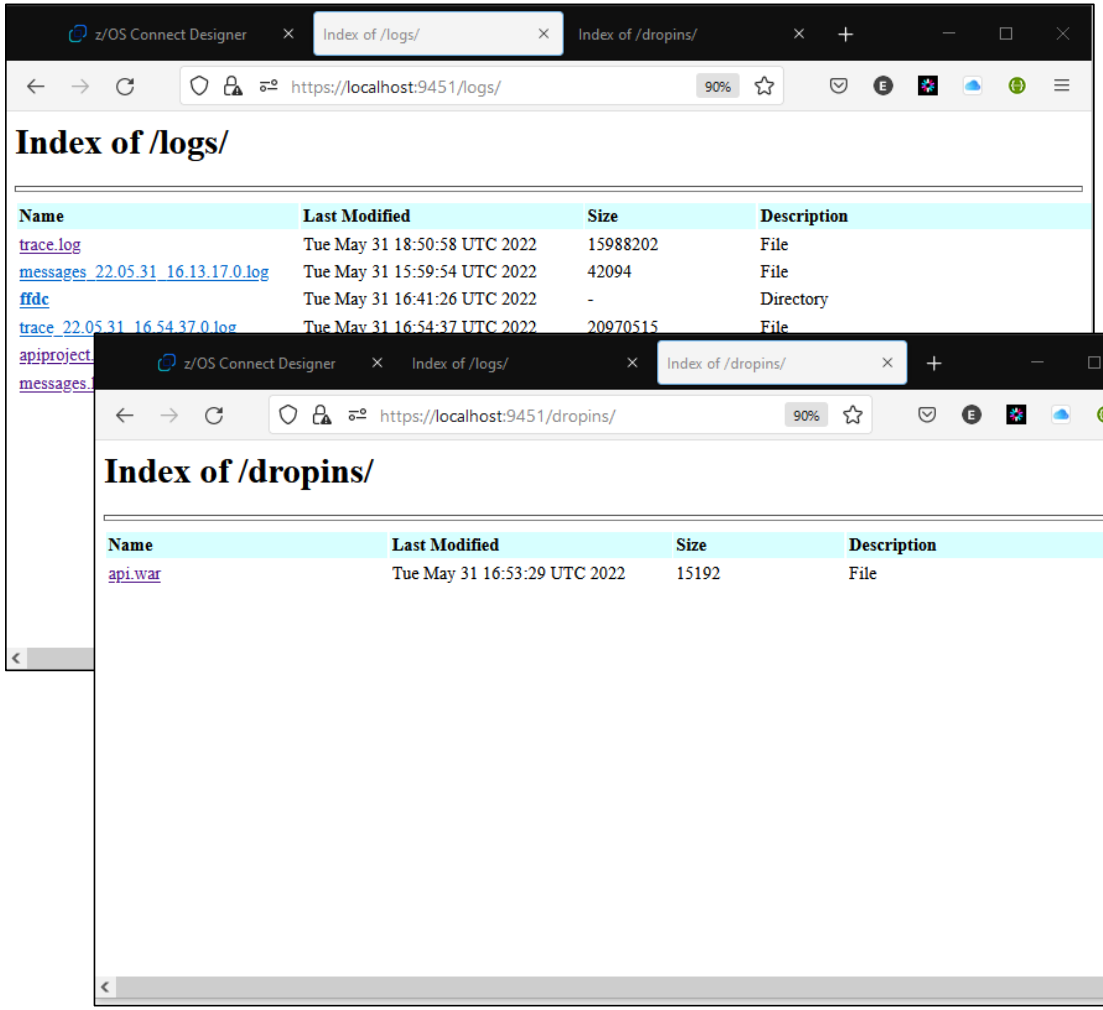
```
REDFINE EJBROLE BBGZDFLT.catalogManager.Manager
REDFINE EJBROLE BBGZDFLT.catalogManager.Staff
```

Access to use the GET method to invoke /items would require read access to EJB role *BBGZDFLT.catalogManager.Manager.*
Access to use the GET method to invoke /items/{id} and the POST method to invoke /orders would require read access to EJB role *BBGZDFLT.catalogManager.Staff.*

mitchj@us.ibm.com

I needed to provide access to the logs and traces and the WAR file from a browser





mitchj@us.ibm.com

Contents of my /mnt/c/z/openApi3/xml/webApplication.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="Default server">
<webApplication id="resources-dropins" name="dropins"
   location="/opt/ibm/wlp/usr/servers/defaultServer/dropins">
    <web-ext context-root="dropins"
      enable-file-serving="true" enable-directory-browsing="true">
      <file-servering-attribute name="enxtendDocumentRoot"
       value="/opt/ibm/wlp/usr/servers/defaultServer/dropins" />
    </web-ext>
</webApplication> >
<webApplication id="resources-logs" name="logs"
   location="/logs">
    <web-ext context-root="logs"
      enable-file-serving="true" enable-directory-browsing="true">
      <file-servering-attribute name="enxtendDocumentRoot"
       value="/logs" />
    </web-ext>
</webApplication> >
</server>
```

mitchj@us.ibm.com

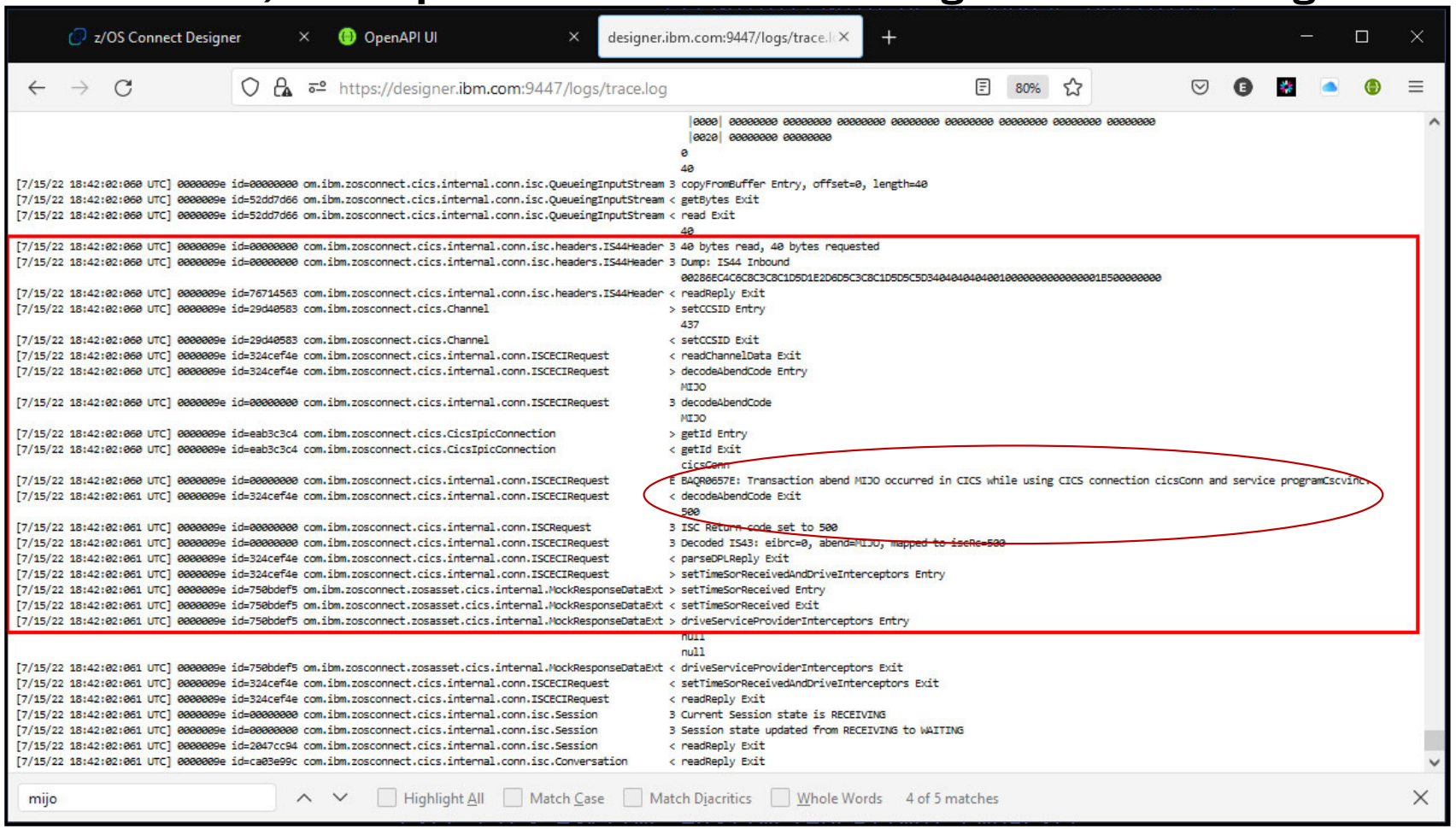
Contents of my /mnt/c/z/openApi3/xml/designerTrace.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="trace specifications">

<logging traceSpecification="
         com.ibm.ws.security.*=all:
         zosConnectCics=all:
         zosConnectDb2=all:
         zosConnectIms=all:
         SSLChannel=all:
         SSL=all:
         "/>
</server>
```

mitchj@us.ibm.com

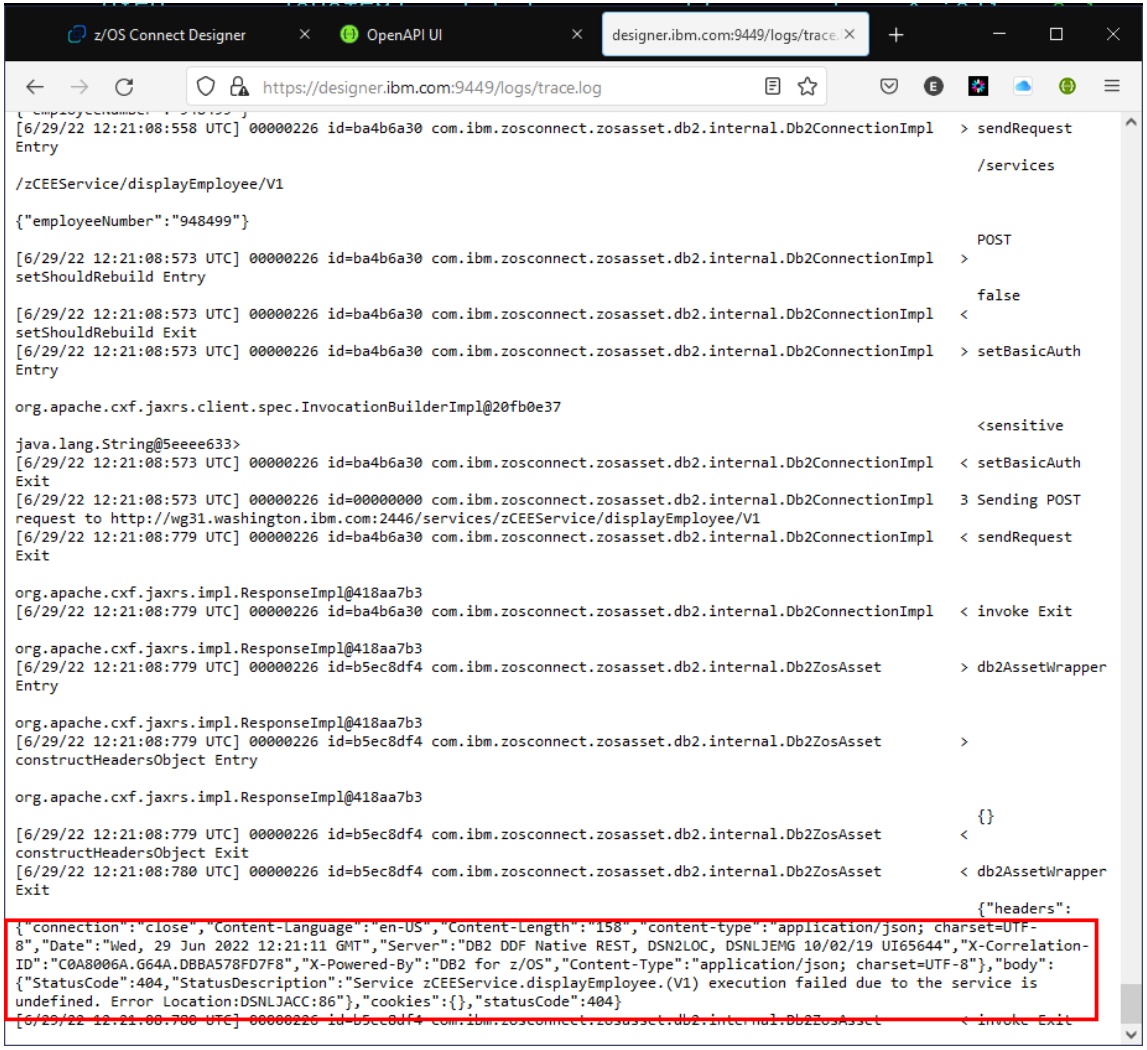
For example, if the Designer displays an HTTP 500 return code

in the same browser, I can open a tab and review the log and trace messages



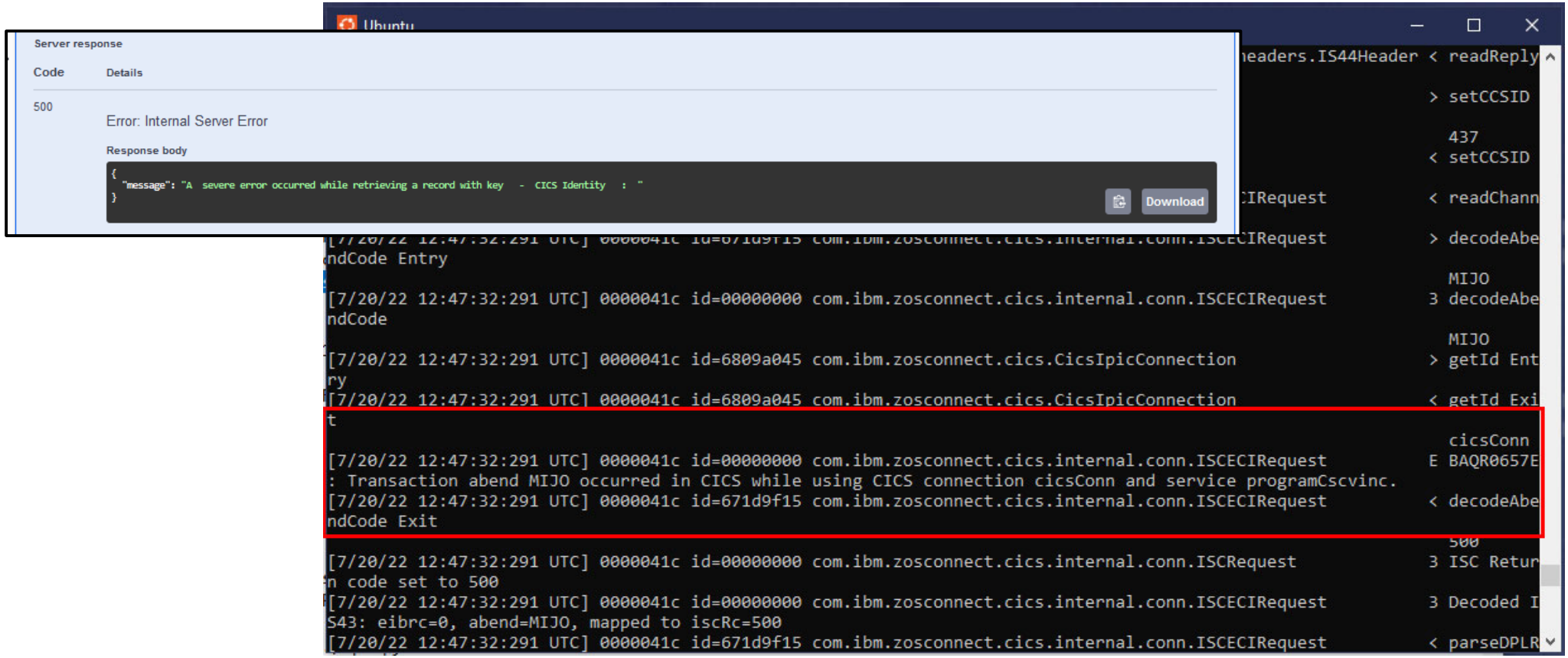
mitchj@us.ibm.com

Of if accessing Db2, I can see the original Db2 SQL error or REST error



mitchj@us.ibm.com

Tech-Tip: Tracing using the tail command *tail –f logs/trace.out &*



- Real time monitoring of the trace output → ***tail –f logs/trace.out &***

mitchj@us.ibm.com

Tech-Tip: Updating and Moving a container to an air gapped image

- Pull in a new (download) a z/OS Connect Designer image
  docker pull icr.io/zosconnect/ibm-zcon-designer:3.0.59
- Save the z/OS Connect Docker image to a file
  docker save icr.io/zosconnect/ibm-zcon-designer:3.0.59 \| gzip > 3.0.59.tar.gz
- Copy the z/OS Connect Docker image file to a base image directory location
  cp *.tar.gz /mnt/c/z/ftp
- Use FTP to move the image file from the original image to the target Linux image
- Load the z/OS Connect Docker image on the Linux image
  docker load < 3.0.57.tar.gz

mitchj@us.ibm.com

Commands for managing the container certificates and key stores

Note my use of *docker exec -it sandbox_zosConnect_1* rather than
docker run -it --rm -v /Users/<username>/Desktop/ZCWorkspace/certs:/tmp/cert/output icr.io/zosconnect/ibm-zcon-designer:3.0.56 as documented at **https://www.ibm.com/docs/en/zos-connect/zos-connect/3.0?topic=db2-configuring-connection-basic-authentication-tls**

- Import the CICS public certificate authority certificate into the local keystore.
  docker exec -it sandbox_zosConnect_1 keytool -importcert -file /output/resources/security/CICSCA.pem -nopromt -keystore /output/resources/security/zosConnect.jks -storetype PKCS12 –alias cicsca

- Import the Db2 public certificate authority certificate into the local keystore.
  docker exec -it sandbox_zosConnect_1 keytool -importcert -file /output/resources/security/DB2CA.pem -nopromt -keystore /output/resources/security/zosConnect.jks -storetype PKCS12 –alias db2ca

- List the contents of the local keystore
  docker exec -it sandbox_zosConnect_1 keytool -v -list -keystore /output/resources/security/zosConnect.jks -storetype PKCS12

mitchj@us.ibm.com

Commands for managing the container certificates and key stores

- Create a self-signed certificate (and create a local keystore)
- *docker exec -it sandbox_zosConnect_1 keytool -keystore /output/resources/security/cicsKeyStore.jks -storetype PKCS12 -storepass changeit -genkey -keysize 2048 -alias cicsusr -dname "CN=user1, O=IBM, C=US" -keyalg RSA -validity 365*

- Create a certificate request from the self-signed certificate

docker exec -it sandbox_zosConnect_1 keytool -keystore /output/resources/security/cicsKeyStore.jks -storetype PKCS12 -certreq -alias cicsusr -file /output/resources/security/user1.arm

- Send the certificate request to the certificate authority for signing

- Import the signed personal certificate into the local key store.
- *docker exec -it cscvinc_zosConnect_1 keytool -importcert -file /output/resources/security/user1.PEM -alias cicsusr -storetype PKCS12 --noprompt -keystore /output/resources/security/cicsKeyStore.jks*

mitchj@us.ibm.com

My TLS *docker-compose.yaml* file

```
version: "3.2"
services:
    zosConnect:
        image: icr.io/zosconnect/ibm-zcon-designer:3.0.57
        user: root
        environment:
            - CICS_USER=USER1
            - CICS_PASSWORD=USER1
            - CICS_HOST=wg31.washington.ibm.com
            - CICS_PORT=1491
            - CICSTRUSTSTORE_PASSWORD=changeit
            - CICSKEYSTORE_PASSWORD=secret
            - CICSSSL_PORT=1493
            - DB2_USERNAME=USER1
            - DB2_PASSWORD=USER1
            - DB2_HOST=wg31.washington.ibm.com
            - DB2_PORT=2446
            - DB2TRUSTSTORE_PASSWORD=changeit
            - Db2KEYSTORE_PASSWORD=secret
            - DB2SSL_PORT=2445
            - HTTP_PORT=9080
        ports:
            - "9447:9443"
            - "9084:9080"
        volumes:
            - ./project:/workspace/project
            - ./logs/:/logs/
            - ./certs:/output/resources/security/
```

docker-compose . . .

podman rm . . .
podman-compose . . .

mitchj@us.ibm.com

Contents of my /mnt/c/z/openApi3/tls/cicsTLSServer.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="IPIC connection to CICS">
    <featureManager>
        <feature>zosconnect:cics-1.0</feature>
    </featureManager>
    <zosconnect_cicsIpicConnection id="cicsConn"
      host="${CICS_HOST}" port="${CICSSSL_PORT}"
      authDataRef="cicsCredentials"
      sslCertsRef=cicsSSLSettings" />
   <ssl id="cicsSSLSettings"
     keyStoreRef= "cicsTrustStore"
     trustStoreRef= "cicsTrustStore" />
   <keyStore id= "cicsTrustStore"
     location="/output/resources/security/cicsTrustStore.jks"
     password="${CICSTRUSTSTORE_PASSWORD}" type="PKCS12" />
    <zosconnect_authData id="cicsCredentials"
     user="${CICS_USER}" password="${CICS_PASSWORD}" />
</server>
```

mitchj@us.ibm.com

Contents of my /mnt/c/z/openApi3/tls/cicsTLSMutual.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="IPIC connection to CICS">
    <featureManager>
        <feature>zosconnect:cics-1.0</feature>
    </featureManager>
    <zosconnect_cicsIpicConnection id="cicsConn"
        host="${CICS_HOST}"
        port="${CICSSSL_PORT}"
        zosConnectNetworkid="DESIGNER"
        zosConnectApplid="DESIGNER"
        sslCertsRef="cicsSSLSettings" />
<ssl id="cicsSSLSettings"
     keyStoreRef= "cicsKeyStore"
     trustStoreRef= "cicsTrustStore" />
   <keyStore id= "cicsTrustStore"
     location="/output/resources/security/cicsTrustStore.jks"
     password="${CICSTRUSTSTORE_PASSWORD}" type="PKCS12" />
   <keyStore id= "cicsKeyStore"
     location="/output/resources/security/CICSUSR1.P12"
     password="${CICSKEYSTORE_PASSWORD}" type="PKCS12" />
</server>
```

mitchj@us.ibm.com

Contents of my /mnt/c/z/openApi3/tls/db2TLSServer.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="DB2 SSL">
    <featureManager>
        <feature>zosconnect:db2-1.0</feature>
    </featureManager>
    <zosconnect_credential id="commonCredentials" />
      user="${DB2_USERNAME}" password="${DB2_PASSWORD}" />
    <zosconnect_db2Connection id="db2ConnTLS"
      host="${DB2_HOST}" port="${DB2SSL_PORT}"
      credentialRef="commonCredentials"
      sslCertsRef="db2SSLSettings" />
     <ssl id="db2SSLSettings"
      keyStoreRef="db2TrustStore"
      trustStoreRef="db2TrustStore" />
    <keyStore id="db2TrustStore"
     location="/output/resources/security/db2TrustStore.jks"
     password="${DB2TRUSTSTORE_PASSWORD}" type="PKCS12" />

</server>
```

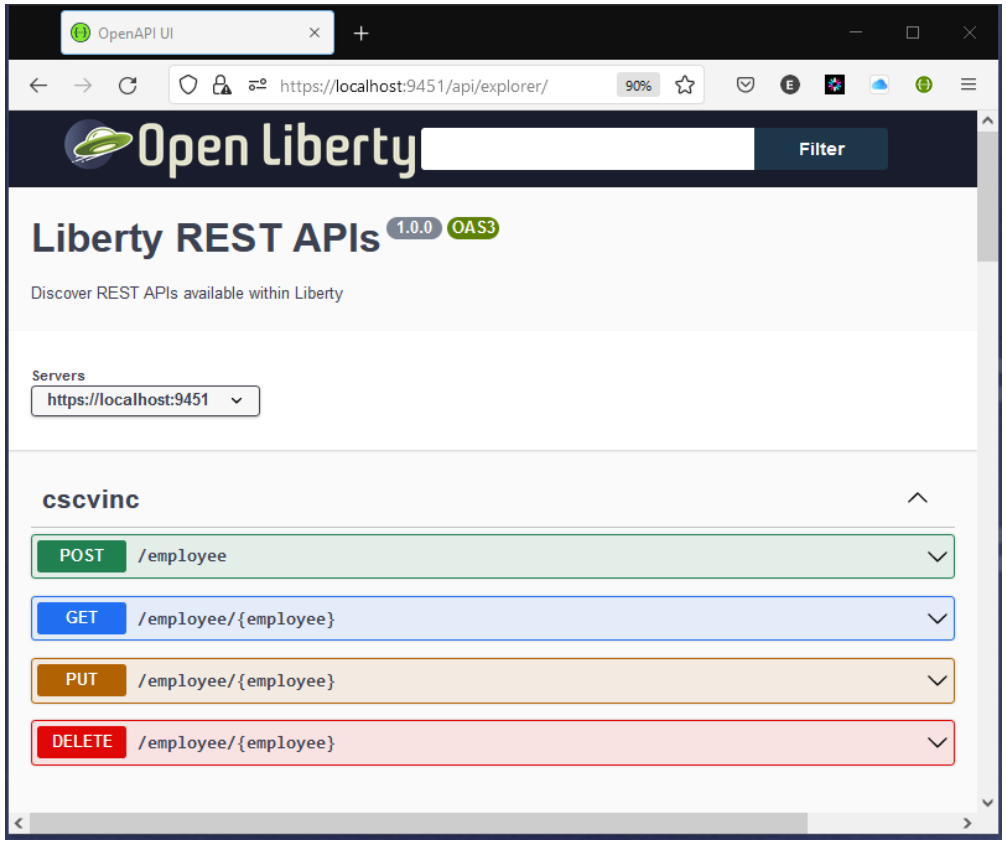
mitchj@us.ibm.com

Useful URLs for z/OS Connect Designer Container

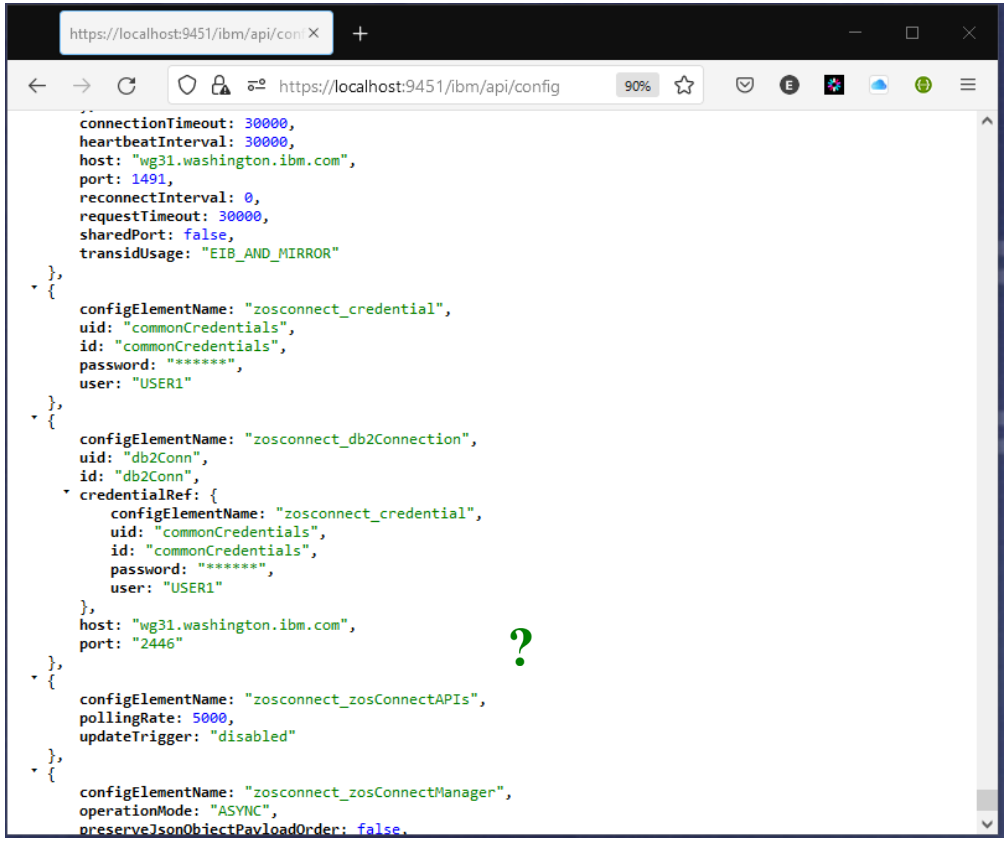
- **Accessing the Designer**
  https://localhost:9445/zosConnect/designer
- **Review the Container's Liberty configuration**
  https://localhost:9445/ibm/api/config
- **Access the Container's API Explorer**
  https://localhost:9445/api/explorer/
- **Access the Container's logs directory**
  https://localhost:9445/logs
- **Access and/or download the Web Archive (WAR) file**
  https://localhost:9445/dropins
- **Convert a Swagger (Open API 2) document to Open API 3**
  https://mermade.org.uk/openapi-converter
- **Validate and/or view an Open API 3 document**
  https://jsonformatter.org/yaml-viewer

mitchj@us.ibm.com

The container's APIs can be tested using a browser (w/o using the Designer)

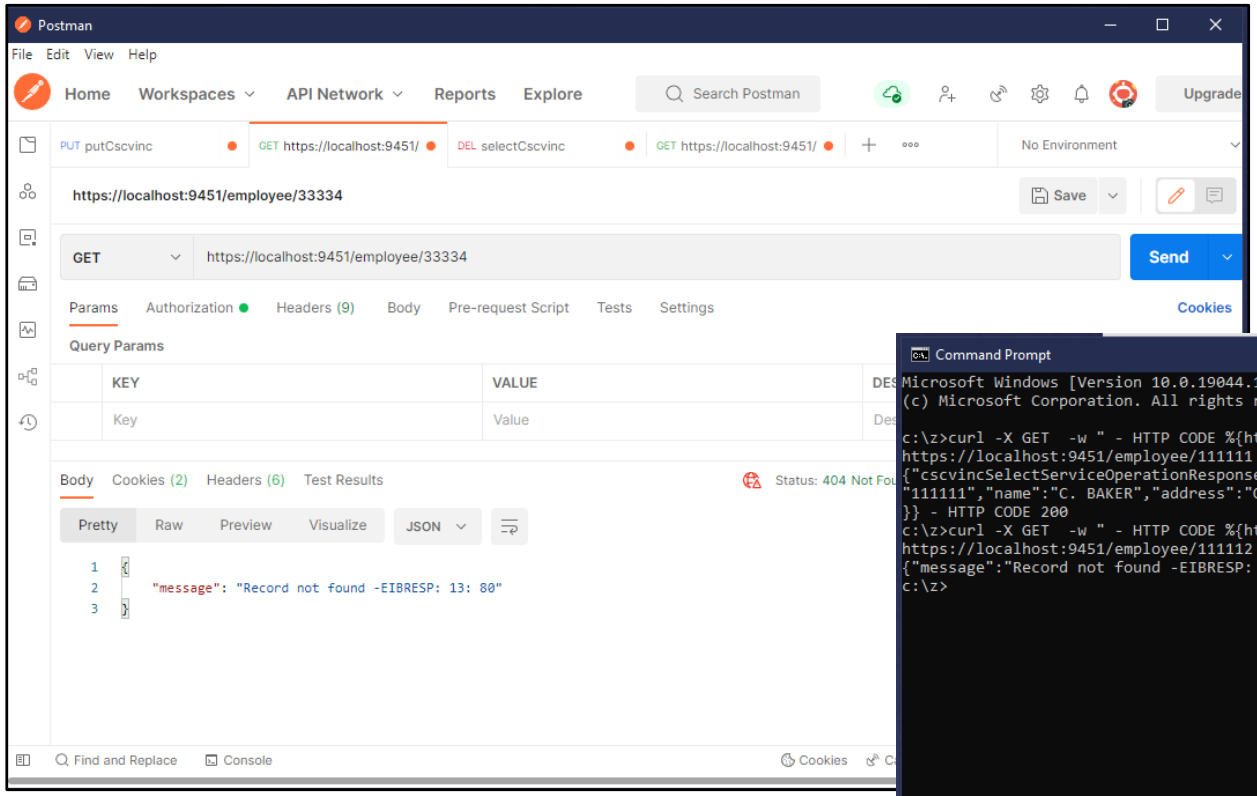


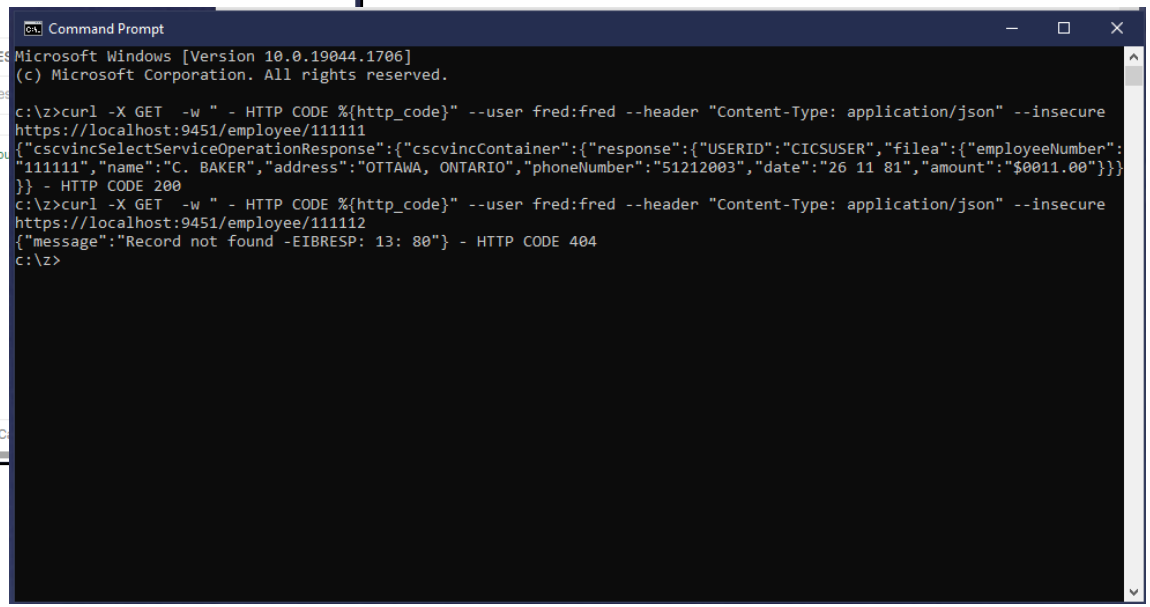
And the configuration can be reviewed



mitchj@us.ibm.com

Remember the container's API can be invoked with Postman or curl





mitchj@us.ibm.com

Thank you for listening and your questions.

mitchj@us.ibm.com

Interesting Liberty defaults for the Designer

```
********************************************************************************
product = WebSphere Application Server 22.0.0.3, z/OS Connect 03.00.57 (wlp-1.0.62.cl220320220302-1100)
wlp.install.dir = /opt/ibm/wlp/
server.output.dir = /opt/ibm/wlp/output/defaultServer/
java.home = /opt/ibm/java/jre
java.version = 1.8.0_321
java.runtime = Java(TM) SE Runtime Environment (8.0.7.6 - pxa6480sr7fp6-20220330_01(SR7 FP6))
os = Linux (5.10.102.1-microsoft-standard-WSL2; amd64) (en_US)
process = 1@192.168.112.2
Classpath = /opt/ibm/wlp/bin/tools/ws-server.jar:/opt/ibm/wlp/bin/tools/ws-javaagent.jar:/opt/ibm/wlp/bin/tools/ws-javaagent.jar
Java Library path = /opt/ibm/java/jre/lib/amd64/compressedrefs:/opt/ibm/java/jre/lib/amd64:/usr/lib64:/usr/lib
********************************************************************************
[6/1/22 18:11:29:925 UTC] 00000001 com.ibm.ws.kernel.launch.internal.FrameworkManager           A CWWKE0001I: The server defaultServer has been
launched.
[6/1/22 18:11:30:827 UTC] 00000027 com.ibm.ws.config.xml.internal.ServerXMLConfiguration       A CWWKG0093A: Processing configuration drop-ins
resource: /opt/ibm/wlp/usr/servers/defaultServer/configDropins/defaults/keystore.xml
[6/1/22 18:11:30:851 UTC] 00000027 com.ibm.ws.config.xml.internal.ServerXMLConfiguration       A CWWKG0093A: Processing configuration drop-ins
resource: /opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides/http-ssl-endpoint.xml
[6/1/22 18:11:30:853 UTC] 00000027 com.ibm.ws.config.xml.internal.ServerXMLConfiguration       A CWWKG0093A: Processing configuration drop-ins
resource: /opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides/tls.xml
[6/1/22 18:11:31:051 UTC] 00000001 com.ibm.ws.kernel.launch.internal.FrameworkManager          I CWWKE0002I: The kernel started after 1.287 seconds
[6/1/22 18:11:31:272 UTC] 00000033 com.ibm.ws.kernel.feature.internal.FeatureManager           I CWWKF0007I: Feature update started.
[6/1/22 18:11:34:054 UTC] 00000027 g.apache.cxf.cxf.core.3.2:1.0.62.cl220320220302-1100(id=90)] I Aries Blueprint packages not available. So
namespaces will not be registered
[6/1/22 18:11:34:118 UTC] 00000026 com.ibm.ws.security.ready.internal.SecurityReadyServiceImpl  I CWWKS0007I: The security service is starting...
[6/1/22 18:11:34:288 UTC] 00000027 com.ibm.ws.app.manager.internal.monitor.DropinMonitor        A CWWKZ0058I: Monitoring dropins for applications.
[6/1/22 18:11:34:788 UTC] 00000027 com.ibm.ws.cache.ServerCache                                 I DYNA1001I: WebSphere Dynamic Cache instance named
baseCache initialized successfully.
[6/1/22 18:11:34:794 UTC] 00000027 com.ibm.ws.cache.ServerCache                                 I DYNA1071I: The cache provider default is being
used.
[6/1/22 18:11:34:796 UTC] 00000027 com.ibm.ws.cache.CacheServiceImpl                            I DYNA1056I: Dynamic Cache (object cache)
initialized successfully.
[6/1/22 18:11:35:004 UTC] 00000026 ibm.ws.security.authentication.internal.jaas.JAASServiceImpl I CWWKS1123I: The collective authentication plugin
with class name NullCollectiveAuthenticationPlugin has been activated.
[6/1/22 18:11:35:425 UTC] 0000004c com.ibm.ws.security.token.ltpa.internal.LTPAKeyCreateTask    I CWWKS4105I: LTPA configuration is ready after
0.455 seconds.
[6/1/22 18:11:35:594 UTC] 00000026 com.ibm.ws.session.WASSessionCore                            I SESN8501I: The session manager did not find a
persistent storage location; HttpSession objects will be stored in the local application server's memory.
[6/1/22 18:11:35:810 UTC] 00000026 .microprofile.metrics.internal.monitor.MonitorMetricsHandler I CWPMI2003I: Monitoring metrics can be retrieved
through mpMetrics.
[6/1/22 18:11:35:911 UTC] 00000026 com.ibm.ws.security.audit.file.AuditFileHandler              I CWWKS5804I: The audit file handler service is
starting.
[6/1/22 18:11:35:916 UTC] 00000026 com.ibm.ws.security.audit.source.AuditServiceImpl            I CWWKS5850I: The audit service is starting.
[6/1/22 18:11:35:924 UTC] 00000026 com.ibm.ws.security.audit.source.AuditServiceImpl            I CWWKS5851I: The audit service is ready.
[6/1/22 18:11:35:930 UTC] 00000026 com.ibm.ws.security.audit.file.AuditFileHandler              I CWWKS5805I: The audit file handler service is
ready.
[6/1/22 18:11:36:195 UTC] 0000002f com.ibm.ws.ssl.config.WSKeyStore                             I Successfully loaded default keystore:
/opt/ibm/wlp/output/defaultServer/resources/security/key.p12 of type: PKCS12
```

/opt/ibm/wlp/usr/servers/defaultServer/server.xml

mitchj@us.ibm.com

Default server XML configuration files

defaults/keystore.xml

```
<server description="Default Server">
    <keyStore id="defaultKeyStore" password="IRawdFa2HxS7cte9T8M1MOvpKQzqtYnQRZwKGffp3t0=" />
```

overrides/http-ssl-endpoint.xml

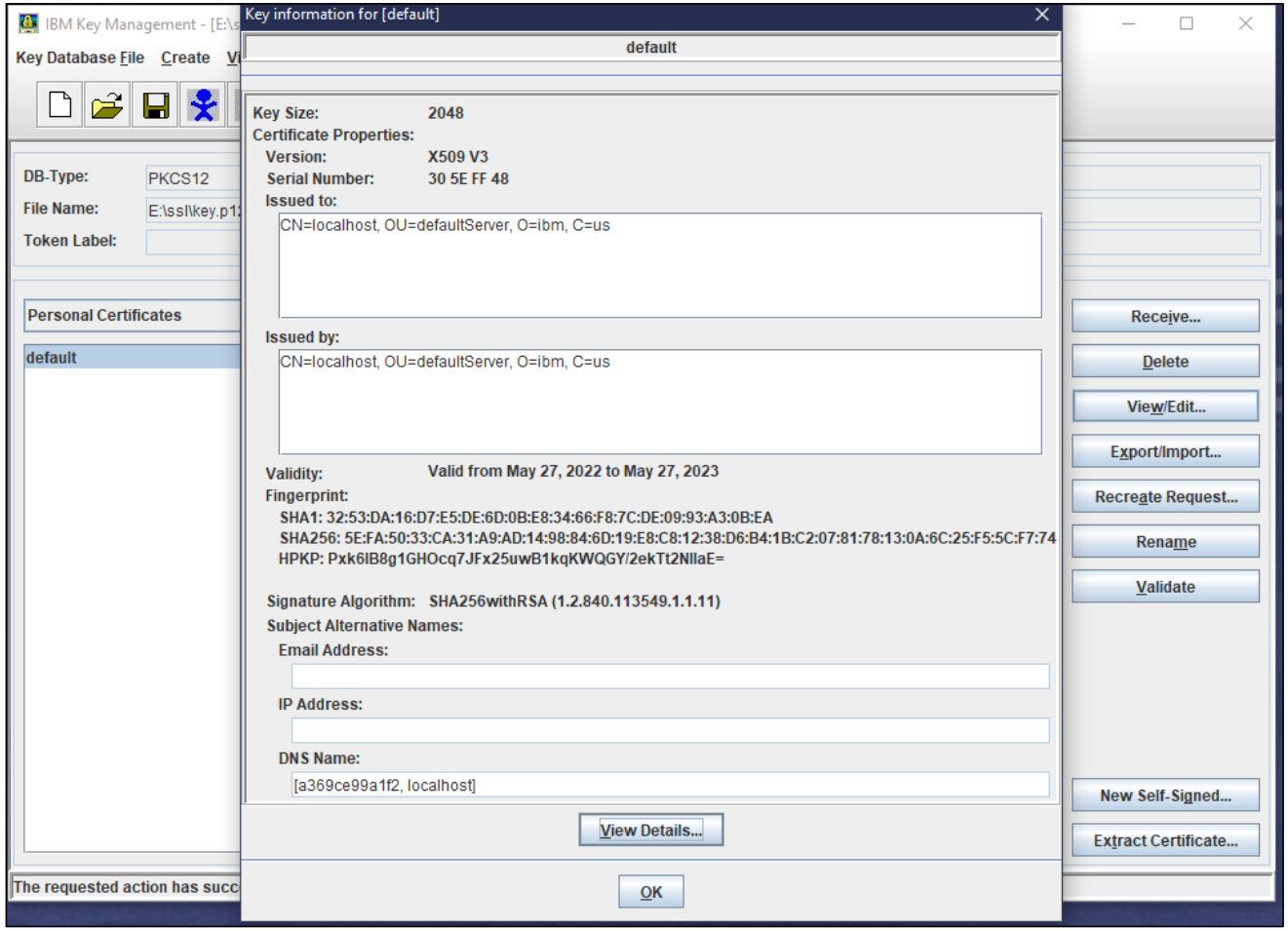
```
<server>
    <variable name="HTTP_PORT" defaultValue="9080" />
    <variable name="HTTPS_PORT" defaultValue="9443" />
    <httpEndpoint id="defaultHttpEndpoint" host="*" httpsPort="${HTTPS_PORT}" httpPort="${HTTP_PORT}" />
</server>
```

overrides/tls.xml

```
<server description="Default Server"
<featureManager>
        <feature>transportSecurity-1.0</features>
</featureManager>
```

mitchj@us.ibm.com

Be wary of the container's default self-signed personal certificate



Note that the certificate expires after 1 year.

In Linux
cp ../sandbox/certs/key.p12 /mnt/c/ssl
In Windows, use ikeyman to open the keystore

mitchj@us.ibm.com

Key directories in the Designer container

- **Key container directories**

  /workspace/project
  /opt/ibm/wlp/usr/servers/defaultServer
  /output/resources/security
  /opt/ibm/wlp/usr/servers/defaultServer/configDropins/default
  /opt/ibm/wlp/usr/servers/defaultServer/configDropins/

  /config -> /opt/ibm/wlp/usr/servers/defaultServer
  /output -> /opt/ibm/wlp/output/defaultServer
  ${server.output.dir} -> /output/ibm/wlp/output/defaultServer

mitchj@us.ibm.com

Useful Windows PowerShell Commands

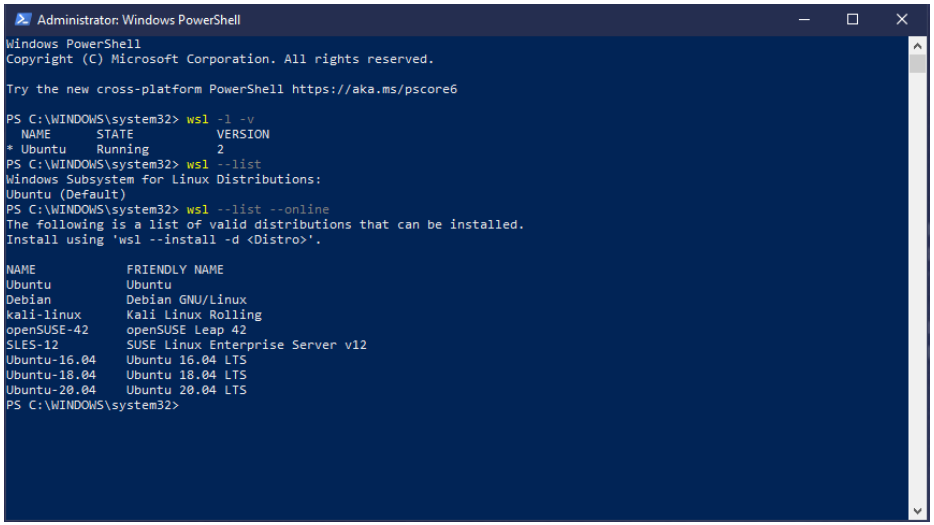
- **WSL Commands**

List details of all distributions
- *wsl -l –v*

List all distributions
- *wsl –list*

Set the default install version for a new distribution
- *wsl --set-default-version 2*

Display a list of available Linux distributions
- *wsl –list --online*



- **Manage Windows Virtualization (requires Administrative authority)**

Enables Windows virtualization (reboot required)
- *bcdedit /set hypervisorlaunchtype auto*

Disables Windows virtualization (reboot required)
- *bcdedit /set hypervisorlaunchtype off*

mitchj@us.ibm.com

Tech-Tip: commands related to creating and managing containers

- Start the docker daemon as a background process (note the use &), ***dockerd &***
- Check to see if the Docker daemon is active or on a Mac that the Podman virtual machine is active.
  ps –ef | grep dockerd
  podman machine init and/or ***podman machine start***

- Start a new container or update an existing container using a *docker-compose-yaml* file
  docker-compose -f /home/workstation/docker/sandbox/docker-compose.yaml up -d
- Start a new container using docker-compose.yaml while in directory /home/workstation/docker/sandbox
  docker-compose up –d
- Stop the container using docker-compose command while in directory /home/workstation/docker/sandbox
  docker-compose down
- Start the sandbox container regardless of current directory
  docker start sandbox_zosconnect_1
- Stop the sandbox container regardless of current directory
  docker stop sandbox_zosconnect_1
- Copy server XML override files from a Windows directory into a container's directory*
  docker cp /mnt/c/z/openApi3/xml/. sandbox_zosConnect_1:/config/configDropins/overrides

mitchj@us.ibm.com

Tech-Tip: Commands for managing containers

- List the active containers
  docker ps

```
CONTAINER ID   IMAGE                                          COMMAND                  CREATED          STATUS
PORTS                                                                                  NAMES
97756ede6692   icr.io/zosconnect/ibm-zcon-designer:3.0.55   "/opt/ibm/helpers/ru…"   26 hours ago     Up 26 hours
0.0.0.0:9088->9080/tcp, :::9088->9080/tcp, 0.0.0.0:9429->9443/tcp, :::9429->9443/tcp   employees_zosConnect_1
642f17a4063a   icr.io/zosconnect/ibm-zcon-designer:3.0.55   "/opt/ibm/helpers/ru…"   47 hours ago     Up 20 hours
0.0.0.0:9082->9080/tcp, :::9082->9080/tcp, 0.0.0.0:9445->9443/tcp, :::9445->9443/tcp   sandbox_zosConnect_1
```

- List all active and stopped containers
  docker ps -a

- Remove a container by name or container ID
  docker rm sandbox_zosconnect_1
  or
  docker rm 642f17a4063a

- Invoke a command in the container
  docker exec -it sandbox_zosConnect_1 bash

mitchj@us.ibm.com

Other useful commands

- List the installed images
  docker images

```
REPOSITORY                                TAG       IMAGE ID       CREATED         SIZE
icr.io/zosconnect/ibm-zcon-designer       3.0.57    386f4ac8cbd0   25 hours ago    1.16GB
icr.io/zosconnect/ibm-zcon-designer       3.0.56    cf167f4230b5   6 weeks ago     1.57GB
icr.io/zosconnect/ibm-zcon-designer       3.0.55    be9c9101f533   2 months ago    1.52GB
hello-world                               latest    feb5d9fea6a5   8 months ago    13.3kB
```

- Remove an installed image
  docker rmi icr.io/zosconnect/ibm-zcon-designer:3.0.56

- Invoking Linux commands in the container
  docker exec -it sandbox_zosConnect_1 ls -l /templates/gradleLibs/
  docker exec -it sandbox_zosConnect_1 ls -lR /templates/gradleLibs/com/ibm/zosconnect
  docker exec -it sandbox_zosConnect_1 cd /workspace/project && gradle build --debug

- Install the Podman podman-compose command
  pip install podman-compose

mitchj@us.ibm.com

Other useful container related commands

- Display the details of a container
  docker container inspect sandbox_zosConnect-1
- Create a copy of a container
  docker commit sandbox_zosConnect_1 sandbox_zosconnect_1_Next
- Copy the configuration XML override file from Linux into the container
  docker cp /mnt/c/z/openApi3/xml/. sandbox_zosConnect_1:/config/configDropins/overrides/
- Copy the war files and from the container
  docker cp sandbox_zosConnect_1:/workspace/project/build/libs/api.war /mnt/c/z/openApi3/wars/cscvinc.war
- Copy the configuration XML files from the container into Linux
  docker cp /mnt/c/z/openApi3/xml/. sandbox_zosConnect_1:/config/configDropins/overrides
- Display a docker container's IP information
  docker inspect -f '{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' db2api_zosConnect_1

```
192.168.176.2
```

mitchj@us.ibm.com

Contents of C:/z/openApi3/bin/createPodmanContainer

```
echo on
[ -z "$2" ] && HTTP_port=9080 || HTTP_port=$2
[ -z "$3" ] && HTTPS_port=9443 || HTTPS_port=$3
echo creating container "$1"_zosConnect_1 with HTTP_port="$HTTP_port" and
HTTPS_port="$HTTPS_port"
mkdir $containerHome/podman/"$1"
cd $containerHome/podman/"$1"
mkdir certs
mkdir logs
mkdir -p project/src/main/liberty/config
cp /mnt/c/z/openApi3/xml/* project/src/main/liberty/config
cp /mnt/c/z/openApi3/yaml/docker-compose.yaml .
sed -i "s/9080:9080/$HTTP_port:9080/" docker-compose.yaml
sed -i "s/9443:9443/$HTTPS_port:9443/" docker-compose.yaml
podman-compose up -d
podman cp /mnt/c/z/openApi3/xml/. "$1"_zosConnect_1:/config/configDropins/overrides
```

Used to create a new container, ***createPodmanContainer containerName***

mitchj@us.ibm.com

Contents of C:/z/openApi3/bin/refreshPodmanContainer

```
echo refreshing container "$1"_zosConnect_1
podman stop "$1"_zosConnect_1
podman container rm "$1"_zosConnect_1
cd $containerHome/podman/"$1"
rm -r project/*
mkdir -p project/src/main/liberty/config
cp /mnt/c/z/openApi3/xml/* project/src/main/liberty/config
podman-compose up -d
podman cp /mnt/c/z/openApi3/xml/. "$1"_zosConnect_1:/config/configDropins/overrides
```

Used to refresh an existing container, ***refreshPodmanContainer*** ***containerName***

Contents of C:/z/openApi3/bin/dockerBash

Used to start a Linux shell within a Docker container, ***dockerBash*** ***containerName***

```
docker exec -it "$1"_zosConnect_1 bash
```

Contents of C:/z/openApi3/bin/podmanBash

Used to start a Linux shell within a Podman container, ***podmanBash*** ***containerName***

```
podman exec -it "$1"_zosConnect_1 bash
```

mitchj@us.ibm.com