jenkinlinux@jenkins:~$ kubectl apply -f aspnetcorelinuxdemo-deploy.yaml

error: error parsing aspnetcorelinuxdemo-deploy.yaml: error converting YAML to JSON: yaml: line 11: mapping values are not allowed in this context

error: error parsing aspnetcorelinuxdemo-deploy.yaml: error converting YAML to JSON: yaml: line 10: found character that cannot start any token

Error from server (NotFound): deployments.extensions jenkin

Solution:

Check if you are using tabs for indentation. YAML doesn't allow tabs; it requires spaces.

YAML File:

While executing :Error ValidationError(Deployment): unknown field "selector" in io.k8s.api.apps.v1.Deployment

Resolution : selector was not in child node list , put space before selector, for all such errors this is resolution

similar issue: ValidationError(Deployment): unknown field “spec” in

spec:

selector:

matchLabels:

If you see an error like the following, you probably didn’t set your Docker context correctly:

COPY failed: stat /var/lib/docker/tmp/docker-builder716498213/WiscoIpsum/WiscoIpsum.csproj: no such file or directory

When building Docker images, Docker copies files from your computer. The relative location of those files is based on the Docker context. You set the context when you run your build command. When Visual Studio generates a Docker file for you, it assumes that you’re going to be using the directory containing your solution (\*.sln) file as your context. When you run a docker build or az acr build, ensure the context is set to the solution root directory.

To set the context correctly, use the --file parameter to point to your Dockerfile and then set the context. In the command above, the context is “.”, which is the current directory.

There are 17 steps in the process, so you’ll see quite a bit of output while the command is running. If the process completed successfully you should see a final line of output similar to the following:

**---------------------------------When running Jenkins Error------------------------**

Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post

Even Same occurred when building Docker build from linux command prompt

Solution:

sudo chmod 666 /var/run/docker.sock

OR

sudo chmod 777 /var/run/docker.sock

Building in workspace C:\Program Files (x86)\Jenkins\workspace\test

[test] $ sh -xe C:\Windows\TEMP\hudson6299483223982766034.sh

The system cannot find the file specified

FATAL: L'exécution de la commande a échoué.

java.io.IOException: Cannot run program "sh" (in directory "C:\Program Files (x86)\Jenkins\workspace\test"): CreateProcess error=2, Le fichier spécifié est introuvable

at java.lang.ProcessBuilder.start(Unknown Source)

at hudson.Proc$LocalProc.<init>(Proc.java:244)

Solution:

This is happens because Jenkins is not aware about the shell path. In Manage Jenkins -> Configure System -> Shell, set the shell path as

C:\Windows\system32\cmd.exe

**Docker Building Docker Container Imgae:**

Refer: https://github.com/radhegithub/aspnetcorelinuxdemo.git

Step 7/17 : COPY ["aspnetcorelinuxdemo/aspnetcorelinuxdemo.csproj", "aspnetcorelinuxdemo/"]

COPY failed: stat /var/lib/docker/tmp/docker-builder729947944/aspnetcorelinuxdemo/aspnetcorelinuxdemo.csproj: no such file or directory

Build step 'Execute shell' marked build as failure

Changed Dockerfile Content as:Original

FROM mcr.microsoft.com/dotnet/core/aspnet:3.1-buster-slim AS base

WORKDIR /app

EXPOSE 80

EXPOSE 443

FROM mcr.microsoft.com/dotnet/core/sdk:3.1-buster AS build

WORKDIR /src

COPY ["aspnetcorelinuxdemo/aspnetcorelinuxdemo.csproj", "aspnetcorelinuxdemo/"]

RUN dotnet restore "aspnetcorelinuxdemo/aspnetcorelinuxdemo.csproj"

COPY . .

WORKDIR "/src/aspnetcorelinuxdemo"

RUN dotnet build "aspnetcorelinuxdemo.csproj" -c Release -o /app/build

FROM build AS publish

RUN dotnet publish "aspnetcorelinuxdemo.csproj" -c Release -o /app/publish

FROM base AS final

WORKDIR /app

COPY --from=publish /app/publish .

ENTRYPOINT ["dotnet", "aspnetcorelinuxdemo.dll"]

**Replaced:**

#See https://aka.ms/containerfastmode to understand how Visual Studio uses this Dockerfile to build your images for faster debugging.

FROM mcr.microsoft.com/dotnet/core/aspnet:3.1-buster-slim AS base

WORKDIR /app

EXPOSE 80

EXPOSE 443

FROM mcr.microsoft.com/dotnet/core/sdk:3.1-buster AS build

WORKDIR /src

COPY ["aspnetcorelinuxdemo.csproj", "./"]

RUN dotnet restore "aspnetcorelinuxdemo.csproj"

COPY . ./aspnetcorelinuxdemo

WORKDIR "/src/aspnetcorelinuxdemo"

RUN dotnet build "aspnetcorelinuxdemo.csproj" -c Release -o /app/build

FROM build AS publish

RUN dotnet publish "aspnetcorelinuxdemo.csproj" -c Release -o /app/publish

FROM base AS final

WORKDIR /app

COPY --from=publish /app/publish .

ENTRYPOINT ["dotnet", "aspnetcorelinuxdemo.dll"]