**convert the run time files into a bar**

**or build a bar in the pipeline with the run time files**

**put the bar into a yaml / image to prepare the integration server**

**Fetch the runtime files from a repository.**

**Convert the runtime files into a BAR file using the mqsibar command.**

**Create an ACE integration server configuration YAML with a reference to the BAR file.**

**Deploy the ACE Integration Server on OpenShift.**

The tasks git-clone, build-bar, create-ace-config, and deploy-ace are placeholder task names. You should replace them with the actual tasks which perform those actions.

git-clone - Clones the Git repo containing your project.

build-bar - Builds a BAR file from the project.

create-ace-config - Creates a configuration for the ACE Integration Server using the BAR file.

deploy-ace - Deploys the ACE Integration Server on OpenShift using the configuration.

Please note, these tasks need to be created and defined according to your specific requirements.

Replace your-output.bar, your-project-dir, and your-project-name with the actual names you're going to use.

Please note that the ace-minimal:latest image is a placeholder and should be replaced with the appropriate ACE image that includes the mqsibar command.

docker login cp.icr.io -u cp -p eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJJQk0gTWFya2V0cGxhY2UiLCJpYXQiOjE2Njk3NDI0NDYsImp0aSI6IjNhNDQxMTg2MGI2YTRiZmZhMzY3NDFhOWU2ZTJjN2JkIn0.nUram6byvdEFkr\_jFB0ek2cpr0sV1rWp7ogOhpjqNpY

#docker pull image location 12.0.8.0-r3

https://www.ibm.com/docs/en/app-connect/containers\_cd?topic=obtaining-app-connect-enterprise-server-image-from-cloud-container-registry

docker pull cp.icr.io/cp/appc/ace-server-prod@sha256:c1fac1a80768e26f64d8fff770b10a96b99b2bd4df2e2cd533e96f1a85ebec5c

Re-tag the image appropriately and then push it to your preferred registry. ( commit it to our gitlab)

docker tag original-image:tag your-registry/your-repo/new-image:tag

Push the image to your registry:

docker push your-registry/your-repo/new-image:tag

You can run the image by using an example command such as:

docker run --name aceserver -p 7600:7600 -p 7800:7800 -p 7843:7843 --env LICENSE=accept --env ACE\_SERVER\_NAME=ACESERVER cp.icr.io/cp/appc/ace-server-prod@sha256:2771ba69f5871c37979576389f5a2040213b913f4d46bfbcafdabdea808f5bad

If you have an ACE server image from IBM Cloud Container Registry, you'd want to pull, re-tag, and then push it to your preferred Docker registry (e.g., GitLab's Docker registry). These operations are typically done outside of the Tekton pipeline, as a setup phase. Here's how you can do it:

bash

# Pull the ACE server image from IBM Cloud Container Registry

docker pull cp.icr.io/cp/appc/ace-server-prod@sha256:c1fac1a80768e26f64d8fff770b10a96b99b2bd4df2e2cd533e96f1a85ebec5c

# Re-tag the image with your preferred tag and push it to your preferred Docker registry

docker tag cp.icr.io/cp/appc/ace-server-prod@sha256:c1fac1a80768e26f64d8fff770b10a96b99b2bd4df2e2cd533e96f1a85ebec5c your-registry/your-repo/ace-server-prod:your-tag

docker push your-registry/your-repo/ace-server-prod:your-tag

Replace your-registry, your-repo, and your-tag with your GitLab Docker registry domain, your repository name, and your preferred tag, respectively.

Now, as for your question about the docker run command, it's generally not included in the Tekton pipeline.

In a typical deployment, your Kubernetes or OpenShift environment would run the Docker image as part of a Deployment, StatefulSet, or a similar object. You'd define these objects in your YAML files which you'd apply to your Kubernetes/OpenShift environment.

That said, the actual steps to run the Docker image depend on your deployment strategy. If you're using Tekton to orchestrate your deployment, you might include a Task to apply the Kubernetes YAML files that define how your image should be run

create-ace-config task as follows:

(Look at yaml)

1. \*\*Pull the ACE image\*\* - Since you're already logged into the IBM registry, go ahead and pull the image:

```bash

docker pull cp.icr.io/cp/appc/ace-server-prod@sha256:c1fac1a80768e26f64d8fff770b10a96b99b2bd4df2e2cd533e96f1a85ebec5c

```

2. \*\*Tag the image\*\* - Next, you should tag the image with a name that refers to your Docker registry. Replace `your-registry`, `your-repo`, and `your-tag` with your Docker registry's address, your repository name, and your preferred tag respectively:

```bash

docker tag cp.icr.io/cp/appc/ace-server-prod@sha256:c1fac1a80768e26f64d8fff770b10a96b99b2bd4df2e2cd533e96f1a85ebec5c your-registry/your-repo/ace-server-prod:your-tag

```

3. \*\*Log in to your registry\*\* - Before you can push the image to your registry, you'll need to log in. Replace `your-registry`, `your-username`, and `your-password` with your Docker registry's address, your username, and your password respectively:

```bash

docker login your-registry -u your-username -p your-password

```

4. \*\*Push the image to your registry\*\* - Finally, push the image to your Docker registry:

```bash

docker push your-registry/your-repo/ace-server-prod:your-tag

```

After following these steps, your ACE image should be available in your Docker registry, ready to be used in your project.

Once the Pod is running, you can copy the bar file to the Pod (and thereby to the PVC) with the command oc cp path/to/your/file.bar ace\_app\_example/temporary-pod:/data.

1. **Delete the temporary Pod**

After you've copied the file, you can delete the temporary Pod with the command oc delete pod temporary-pod.

After these steps, you'll have a PVC named bar-file-pvc in the ace\_app\_example namespace, and this PVC will contain your bar file. This PVC can be mounted into your ACE server Pods so that the ACE server can access the bar file.