User Guide

Introduction to Payara Server Docker Nodes & Instances

The Payara® Platform - Production-Ready,
Cloud Native and Aggressively Compatible.

Contents

Introduction & Scope

This guide will detail the basic steps required to get a simple application deployed to a Payara domain with a number of Payara Server instances running in Docker containers and assigned to a deployment group.

This guide won't cover any complicated networking or TLS setups, and doesn't cover the usage of Payara Micro instances.

This guide assumes you have basic understanding of Docker and Payara Platform terminology, but as a quick overview:

- Docker Image – A template for creating containers from
- Docker Container – An instance of a Docker Image
- Payara Server Docker Node / Managed Docker Node – A Docker container and Payara Server instance created and managed by a Payara Server DAS, acting much like a regular Payara Server instance
- Temporary Docker Node / Unmanaged Docker Node – A Docker container and Payara Server instance created separately from a Payara Server DAS, with the Payara Server instance only being registered to the DAS as long as the container is running.
- Temporary Docker Instance / Unmanaged Docker Instance – The same as a Temporary Docker Node, but referring explicitly to the Payara Server instance inside the container.
- DAS – Domain Administration Server of Payara Server
- Asadmin CLI – Payara Server's CLI

Setup

Setup of DAS

This section covers the setup of the Payara DAS, the creation of our shared config and deployment group, and deployment of our application that we want to be available on each of our instances.

Enable Secure Admin

Secure admin is required for remote administration, which will be required if using Payara Server instances in your Docker containers. On the machine that you'll run your DAS from, download and unzip Payara 5.194 and run the following asadmin commands:

1. Start the default domain: `asadmin start-domain`
2. Run the change-admin-password command: `asadmin change-admin-password`
3. Enable secure admin: `asadmin enable-secure-admin`
4. Restart the domain to activate the changes: `asadmin restart-domain`

Create our Instance Config and Deployment Group

Rather than have each instance create its own separate config, we can create a specify a config that they'll share with an altered HTTP port:

```
asadmin copy-config --systemproperties HTTP_LISTENER_PORT=27070 default-config
Gruppy-config
```

To automatically have an application deployed to a selection of instances, we can use a deployment group; any instance that joins it will have any applications targeted at the Deployment Group deployed to them.

```
asadmin create-deployment-group Gruppy1
```

This demo makes use of a simple web application, that exposes a GET and a PUT JAX-RS endpoint, used for retrieving and storing data into a cache respectively. You can download it here. Once you've downloaded it, you can then deploy it to the deployment group with the *deploy* command, making sure to select the pre-created virtual server of *server*:

```
asadmin deploy --target Gruppy1 --virtualservers server rest-jcache.war
```

Initial Setup of Docker Machine

This section covers the setup required of our Docker Image and the machine upon which the Docker containers will run.

Exposing the Docker REST API and Starting the Docker Service

Although not necessary for the creation and usage of unmanaged / temporary Docker instances, we will require the Docker REST API to be exposed for use with the creation of Managed Docker Nodes and instances later on in this example.

Add the following to the Docker start options, on SUSE Linux and Amazon Linux the file can be found under /etc/sysconfig/docker:

```
DOCKER_OPTS="-H 0.0.0.0:2376"
```

Once this has been set, start (or restart) the Docker service:

```
sudo service docker start
```

Since Docker is now no listening on the default Unix socket, you will need to set the DOCKER_HOST variable for the CLI to be able to talk to the Docker daemon:

```
export DOCKER_HOST=localhost:2376
```

Create and Configure Dockerfile

First off we need to create the Dockerfile from which we're going to build our image from. For this example we can simply use the default Dockerfile that is used for Payara Server for the managed Docker Nodes (payara/server-node):

```
FROM azul/zulu-openjdk:8u232
```

```
ENV WORK_DIR=/opt/payara
ENV PAYARA_HOME=${WORK_DIR}/payara5 \
    PAYARA_PASSWORD_FILE_DIR=${WORK_DIR}/passwords \
    PAYARA_DAS_HOST="localhost" \
    PAYARA_DAS_PORT="4848" \
    PAYARA_NODE_NAME="" \
    PAYARA_CONFIG_NAME="" \
    PAYARA_INSTANCE_NAME="" \
    DOCKER_CONTAINER_IP=""
```

```
ENV PAYARA_PASSWORD_FILE=${PAYARA_PASSWORD_FILE_DIR}/passwordfile.txt
```

```
# Create and set the Payara user and working directory owned by the new user
RUN mkdir ${WORK_DIR} && \
    mkdir ${PAYARA_HOME} && \
```

```
    mkdir ${PAYARA_PASSWORD_FILE_DIR} && \
    groupadd -g 1000 payara && \
    useradd -u 1000 -M -s /bin/bash -d ${WORK_DIR} payara -g payara && \
    echo payara:payara | chpasswd && \
    chown -R payara:payara ${WORK_DIR}

USER payara
WORKDIR ${WORK_DIR}

# Install Payara Server and remove unused domains
ARG PAYARA_INSTALL
COPY --chown=payara:payara ${PAYARA_INSTALL} ${PAYARA_HOME}
RUN rm -rf ${PAYARA_HOME}/glassfish/domains/domain1/ && \
    rm -rf ${PAYARA_HOME}/glassfish/domains/production

# Install entrypoint script
ARG ENTRYPOINT_SCRIPT
COPY --chown=payara:payara ${ENTRYPOINT_SCRIPT} ${WORK_DIR}
RUN chmod +x ${WORK_DIR}/entrypoint.sh

# Start the instance
ENTRYPOINT ["/opt/payara/entrypoint.sh"]
```

This Dockerfile creates a working directory under /opt, sets some default environment variables, copies a specified Payara install to it, clears away the domain directories (since we're not using them), and then copies and runs a specified entrypoint script.

Download and Unzip Payara

As noted in the Dockerfile, we have to provide it with a Payara Server install. Download Payara Server 5.194 and unzip it.

Create and Configure Entrypoint Script

We're going to use the default entrypoint script for managed Payara Server Docker Nodes again here, but edit it to check for the existence of an environment variable denoting a deployment group to join. I'll explain the script in sections below:

Initial Setup

This section sets the `DOCKER_CONTINAER_ID and DOCKER_CONTAINER_IP` environment variables, which are used in the script when creating instances to register information about this specific

container to the DAS to facilitate communication between the two. This hasn't been changed from the default entrypoint script.

```
#!/usr/bin/env bash
set -e
```

```
DOCKER_CONTAINER_ID="$(cat /proc/self/cgroup | grep :/docker/  | sed s/\\//\\
n/g | tail -1)"
echo "Docker Container ID is: ${DOCKER_CONTAINER_ID}"
```

```
if [ -z "${DOCKER_CONTAINER_IP}" ]; then
    echo "No Docker container IP override given, setting to first result from
'hostname -I'"
    DOCKER_CONTAINER_IP="$(hostname -I | cut -f1 -d ' ')"
    echo "Hostname is ${DOCKER_CONTAINER_IP}"
fi
```

```
echo "Docker Container IP is: ${DOCKER_CONTAINER_IP}"
```

Check for Managed Docker Node

This function checks whether or not a new temporary, unmanaged node needs to be created for this Docker container, or whether or not this container is hosted on a managed Docker Node registered to the DAS. This hasn't been changed from the default entrypoint script.

This section (and some of the following ones) run an asadmin command against the DAS. Since the DAS is remote, secure admin will have been enabled, mandating the use of a password file.

```
### Functions ###
function checkAndCreateNewNodeIfRequired {
    if [ -z "${PAYARA_NODE_NAME}" ]; then
        echo "No node name given."
        AUTOGENERATE_NODE_NAME=true
        createNewNode
    else
        # Check if node exists and matches this IP address
        echo "Node name provided, checking if node details match this
container."
        NODE_EXISTS="$(./payara5/bin/asadmin -I false -H ${PAYARA_DAS_HOST} -p
${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} get nodes.node.${PAYARA_NODE_
NAME}.name)" || true
```

```
        if [ ! -z "${NODE_EXISTS}" ]; then
            # Cut off the "Command completed succesfully bit
            NODE_EXISTS="$(echo ${NODE_EXISTS} | cut -f1 -d ' ')"
            if [ "${NODE_EXISTS}" == "nodes.node.${PAYARA_NODE_NAME}.
name=${PAYARA_NODE_NAME}" ]; then
                echo "Node with matching name found, checking node details."
                NODE_HOST="$(./payara5/bin/asadmin -I false -H ${PAYARA_DAS_
HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} get nodes.node.${PAYARA_
NODE_NAME}.node-host)" || true
                if [ ! -z "${NODE_HOST}" ]; then
                    # Cut off the "Command completed succesfully bit
                    NODE_HOST="$(echo ${NODE_HOST} | cut -f1 -d ' ')"
                    echo "Node Host of matching node is ${NODE_HOST}"
                    if [ "${NODE_HOST}" == "nodes.node.${PAYARA_NODE_NAME}.
node-host=${DOCKER_CONTAINER_IP}" ]; then
                        echo "Node details match, no need to create a new
node."
                    else
                        echo "Node details do not match, creating a new node."
                        AUTOGENERATE_NODE_NAME=true
                        createNewNode
                    fi
                else
                    echo "Could not retrieve node host, creating a new node."
                    AUTOGENERATE_NODE_NAME=true
                    createNewNode
                fi
            else
                echo "No node with matching name found."
                AUTOGENERATE_NODE_NAME=false
                createNewNode
            fi
        else
            echo "No node with matching name found."
            AUTOGENERATE_NODE_NAME=false
            createNewNode
        fi
    fi
}
```

Create New Node

This function creates a new temporary node for use specifically with this container. It checks for if a node name has actually been provided, generating one as it creates a temporary node if one hasn't. This hasn't been changed from the default entrypoint script.

```
function createNewNode {
    echo "WARNING: Could not find a matching Docker Node: Creating a temporary
node specific to this container - cleanup of this container cannot be done by
Payara Server"
    if [ "${AUTOGENERATE_NODE_NAME}" ]; then
        echo "Creating a temporary node with an autogenerated name."
        ASADMIN_COMMAND="./payara5/bin/asadmin -I false -T -a -H ${PAYARA_
DAS_HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} _create-node-temp
--nodehost ${DOCKER_CONTAINER_IP}"
        echo "${ASADMIN_COMMAND}"
        PAYARA_NODE_NAME="$(${ASADMIN_COMMAND})"
    else
        echo "Creating a temporary node with provided name."
        ASADMIN_COMMAND="./payara5/bin/asadmin -I false -T -H ${PAYARA_DAS_
HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} _create-node-temp
--nodehost ${DOCKER_CONTAINER_IP} ${PAYARA_NODE_NAME}"
        echo "${ASADMIN_COMMAND}"
        PAYARA_NODE_NAME="$(${ASADMIN_COMMAND})"
    fi
}
```

Create New Instance

This function creates new instances, utilising a config, name, or deployment group if any were provided as environment variables. Similar to the temporary node creation, if no name has been provided one will be generated.

This is the only function that we've edited for this example; the edited sections are highlighted. At each point where the `create-local-instance` command is called, a presence check has been added for the `PAYARA_DEPLOYMENT_GROUP` environment variable, and if present, the `--deploymentgroup ${PAYARA_DEPLOYMENT_GROUP}` parameter and operand have been added.

```
function createNewInstance {
    # Check if we actually have a node name. If we don't have a node name, we
can assume that we need to create a node from scratch
    checkAndCreateNewNodeIfRequired
```

```
    echo "Running command create-local-instance:"
    if [ -z "${PAYARA_INSTANCE_NAME}" ]; then
        if [ -z "${PAYARA_CONFIG_NAME}" ]; then
            if [ -z "${PAYARA_DEPLOYMENT_GROUP}" ]; then
                ASADMIN_COMMAND="./payara5/bin/asadmin -I false -T -a -H
${PAYARA_DAS_HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} create-
local-instance --node ${PAYARA_NODE_NAME} --dockernode true --ip ${DOCKER_
CONTAINER_IP}"
                echo "${ASADMIN_COMMAND}"
                PAYARA_INSTANCE_NAME="$(${ASADMIN_COMMAND})"
            else
                ASADMIN_COMMAND="./payara5/bin/asadmin -I false -T -a -H
${PAYARA_DAS_HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} create-
local-instance --node ${PAYARA_NODE_NAME} --dockernode true --ip ${DOCKER_
CONTAINER_IP} --deploymentgroup ${PAYARA_DEPLOYMENT_GROUP}"
                echo "${ASADMIN_COMMAND}"
                PAYARA_INSTANCE_NAME="$(${ASADMIN_COMMAND})"
            fi
        else
            if [ "${PAYARA_CONFIG_NAME}" == "server-config" ] || [ "${PAYARA_
CONFIG_NAME}" == "default-config" ]; then
                echo "You cannot use 'server-config' or 'default-config',
ignoring provided config name."
                if [ -z "${PAYARA_DEPLOYMENT_GROUP}" ]; then
                    ASADMIN_COMMAND="./payara5/bin/asadmin -I false -T -a -H
${PAYARA_DAS_HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} create-
local-instance --node ${PAYARA_NODE_NAME} --dockernode true --ip ${DOCKER_
CONTAINER_IP}"
                    echo "${ASADMIN_COMMAND}"
                    PAYARA_INSTANCE_NAME="$(${ASADMIN_COMMAND})"
                else
                    ASADMIN_COMMAND="./payara5/bin/asadmin -I false -T -a -H
${PAYARA_DAS_HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} create-
local-instance --node ${PAYARA_NODE_NAME} --dockernode true --ip ${DOCKER_
CONTAINER_IP} --deploymentgroup ${PAYARA_DEPLOYMENT_GROUP}"
                    echo "${ASADMIN_COMMAND}"
                    PAYARA_INSTANCE_NAME="$(${ASADMIN_COMMAND})"
                fi
            else
                if [ -z "${PAYARA_DEPLOYMENT_GROUP}" ]; then
                    ASADMIN_COMMAND="./payara5/bin/asadmin -I false -T -a -H
${PAYARA_DAS_HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} create-
```

```
local-instance --node ${PAYARA_NODE_NAME} --config ${PAYARA_CONFIG_NAME}
--dockernode true --ip ${DOCKER_CONTAINER_IP}"
                    echo "${ASADMIN_COMMAND}"
                    PAYARA_INSTANCE_NAME="$(${ASADMIN_COMMAND})"
                else
                    ASADMIN_COMMAND="./payara5/bin/asadmin -I false -T -a -H
${PAYARA_DAS_HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} create-
local-instance --node ${PAYARA_NODE_NAME} --config ${PAYARA_CONFIG_NAME}
--dockernode true --ip ${DOCKER_CONTAINER_IP} --deploymentgroup ${PAYARA_
DEPLOYMENT_GROUP}"
                    echo "${ASADMIN_COMMAND}"
                    PAYARA_INSTANCE_NAME="$(${ASADMIN_COMMAND})"
                fi
            fi
        fi
    else
        if [ -z "${PAYARA_CONFIG_NAME}" ]; then
            if [ -z "${PAYARA_DEPLOYMENT_GROUP}" ]; then
                ASADMIN_COMMAND="./payara5/bin/asadmin -I false -T -a -H
${PAYARA_DAS_HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} create-
local-instance --node ${PAYARA_NODE_NAME} --dockernode true --ip ${DOCKER_
CONTAINER_IP} ${PAYARA_INSTANCE_NAME}"
                echo "${ASADMIN_COMMAND}"
                PAYARA_INSTANCE_NAME="$(${ASADMIN_COMMAND})"
            else
                ASADMIN_COMMAND="./payara5/bin/asadmin -I false -T -a -H
${PAYARA_DAS_HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} create-
local-instance --node ${PAYARA_NODE_NAME} --dockernode true --ip ${DOCKER_
CONTAINER_IP} ${PAYARA_INSTANCE_NAME} --deploymentgroup ${PAYARA_DEPLOYMENT_
GROUP}"
                echo "${ASADMIN_COMMAND}"
                PAYARA_INSTANCE_NAME="$(${ASADMIN_COMMAND})"
            fi
        else
            if [ "${PAYARA_CONFIG_NAME}" == "server-config" ] || [ "${PAYARA_
CONFIG_NAME}" == "default-config" ]; then
                if [ -z "${PAYARA_DEPLOYMENT_GROUP}" ]; then
                    echo "You cannot use 'server-config' or 'default-config',
ignoring provided config name."
                    ASADMIN_COMMAND="./payara5/bin/asadmin -I false -T -a -H
${PAYARA_DAS_HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} create-
local-instance --node ${PAYARA_NODE_NAME} --dockernode true --ip ${DOCKER_
CONTAINER_IP} ${PAYARA_INSTANCE_NAME}"
```

```
                    echo "${ASADMIN_COMMAND}"
                    PAYARA_INSTANCE_NAME="$(${ASADMIN_COMMAND})"
                else
                    echo "You cannot use 'server-config' or 'default-config',
ignoring provided config name."
                    ASADMIN_COMMAND="./payara5/bin/asadmin -I false -T -a -H
${PAYARA_DAS_HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} create-
local-instance --node ${PAYARA_NODE_NAME} --dockernode true --ip ${DOCKER_
CONTAINER_IP} ${PAYARA_INSTANCE_NAME} --deploymentgroup ${PAYARA_DEPLOYMENT_
GROUP}"
                    echo "${ASADMIN_COMMAND}"
                    PAYARA_INSTANCE_NAME="$(${ASADMIN_COMMAND})"
                fi
            else
                if [ -z "${PAYARA_DEPLOYMENT_GROUP}" ]; then
                    ASADMIN_COMMAND="./payara5/bin/asadmin -I false -T -a -H
${PAYARA_DAS_HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} create-
local-instance --node ${PAYARA_NODE_NAME} --config ${PAYARA_CONFIG_NAME}
--dockernode true --ip ${DOCKER_CONTAINER_IP} ${PAYARA_INSTANCE_NAME}"
                    echo "${ASADMIN_COMMAND}"
                    PAYARA_INSTANCE_NAME="$(${ASADMIN_COMMAND})"
                else
                    ASADMIN_COMMAND="./payara5/bin/asadmin -I false -T -a
-H ${PAYARA_DAS_HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE}
create-local-instance --node ${PAYARA_NODE_NAME} --config ${PAYARA_CONFIG_
NAME} --dockernode true --ip ${DOCKER_CONTAINER_IP} ${PAYARA_INSTANCE_NAME}
--deploymentgroup ${PAYARA_DEPLOYMENT_GROUP}"
                    echo "${ASADMIN_COMMAND}"
                    PAYARA_INSTANCE_NAME="$(${ASADMIN_COMMAND})"
                fi
            fi
        fi
    fi
```

```
    # Register Docker container ID to DAS
    echo "Setting Docker Container ID for instance ${PAYARA_INSTANCE_NAME}:
${DOCKER_CONTAINER_ID}"
    ASADMIN_COMMAND="./payara5/bin/asadmin -I false -H ${PAYARA_DAS_HOST}
-p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} _set-docker-container-id
--instance ${PAYARA_INSTANCE_NAME} --id ${DOCKER_CONTAINER_ID}"
    echo "${ASADMIN_COMMAND}"
```

```
    ${ASADMIN_COMMAND}
}
```

Setup Container

This section is where the previous functions are used, as this is where various checks about the state of the container itself and the environment variables provided are done, determining if a node and/or instance are required to be created.

```
### Setup ###
```

```
# Check if we actually have an instance name. If we don't have an instance
name, we can assume that we need to create an instance from scratch
if [ -z "${PAYARA_INSTANCE_NAME}" ]; then
    echo "No Instance name given."
    createNewInstance
else
    # Check if instance already created before running create command
    if [ ! -d "payara5/glassfish/nodes/${PAYARA_NODE_NAME}/${PAYARA_INSTANCE_
NAME}" ]; then
        echo "Instance name provided, but local file system for instance
missing, checking if file system or new instance needs to be created."
```

```
        # Check if an instance with this name is actually registered
        echo "Checking if an instance with name ${PAYARA_INSTANCE_NAME} has
been registered with the DAS"
        ASADMIN_COMMAND="./payara5/bin/asadmin -I false -t -H ${PAYARA_
DAS_HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} list-instances
--nostatus ${PAYARA_INSTANCE_NAME}"
        echo "${ASADMIN_COMMAND}"
        INSTANCE_EXISTS="$(${ASADMIN_COMMAND})" || true
```

```
        if [ ! -z "${INSTANCE_EXISTS}" ]; then
            # Check if Docker container ID registered against the instance name
is the same
            echo "Found an instance with name ${PAYARA_INSTANCE_NAME}
registered to the DAS, checking if registered Docker Container ID matches this
container's ID"
```

```
            ASADMIN_COMMAND="./payara5/bin/asadmin -I false -t -H ${PAYARA_DAS_
HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} _get-docker-container-id
--instance ${PAYARA_INSTANCE_NAME}"
            echo "${ASADMIN_COMMAND}"
            REGISTERED_DOCKER_CONTAINER_ID="$(${ASADMIN_COMMAND})" || true
```

```
            if [ ! -z "${REGISTERED_DOCKER_CONTAINER_ID}" ]; then
                # If they're the same, simply create the folders, otherwise
create and register a new instance
                echo "Registered Docker Container ID is: ${REGISTERED_DOCKER_
CONTAINER_ID}"
                if [ "${REGISTERED_DOCKER_CONTAINER_ID}" == "${DOCKER_
CONTAINER_ID}" ]; then
                    echo "Docker Container IDs match, creating local instance
filesystem: "
                    ASADMIN_COMMAND="./payara5/bin/asadmin -I false -T -H
${PAYARA_DAS_HOST} -p ${PAYARA_DAS_PORT} -W ${PAYARA_PASSWORD_FILE} _create-
instance-filesystem --node ${PAYARA_NODE_NAME} --dockernode true ${PAYARA_
INSTANCE_NAME}"
                    ${ASADMIN_COMMAND}
                else
                    echo "Docker Container IDs do not match, creating a new
instance."
                    createNewInstance
                fi
            else
                echo "Could not retrieve registered Docker Container ID,
creating a new instance"
                createNewInstance
            fi
        else
            createNewInstance
        fi
    fi
fi
```

Start the Payara Server Instance

The final part of the script simply starts the instance.

```
### Start ###
echo "Starting instance ${PAYARA_INSTANCE_NAME}"
ASADMIN_COMMAND="./payara5/bin/asadmin --passwordfile ${PAYARA_PASSWORD_FILE}
start-local-instance --node ${PAYARA_NODE_NAME} --verbose ${PAYARA_INSTANCE_
NAME}"
${ASADMIN_COMMAND}
```

Create Password File

As noted above, the entrypoint script runs asadmin commands against the DAS which will have secure admin enabled, mandating the use of a password file since this won't be an interactive environment from which the CLI can prompt for the password. Create a file named *passwordfile.txt* and fill it with the following, replacing *${password}* with the password you created earlier:

```
AS_ADMIN_PASSWORD=${password}
```

Create and Tag Docker Image

Now that we have our Dockerfile, entrypoint script, unzipped Payara Server, and passwordfile, we can create and tag the Docker image. We need to pass in the build arguments of our Payara Server installation and entrypoint script, and tag it for ease of use so we can refer to the image via name rather than by ID.

```
docker build --build-arg PAYARA_INSTALL=payara5 --build-arg ENTRYPOINT_
SCRIPT=entrypoint.sh -t cdl-demo .
```

The above command assumes all of the previously listed files are in the same directory.

Creating and Using Temporary / Unmanaged Docker Instances

Temporary Docker instances are Payara Server instances that have been created specifically for the purpose of existing only for the lifetime of the Docker Container – the intention being that these instances will be dynamically added to and removed from an existing Payara Server domain by an orchestrator such as Kubernetes as it spins up or down Docker container. For this example however, we'll simply use the Docker CLI and REST API.

Creating the Container

First, run the following Docker CLI command to create a container, replacing *${path}* with the absolute path to where you've stored the password file created earlier, and *${PAYARA_DAS_HOST}* with the IP address or hostname of the machine that the DAS is running on:

```
sudo docker container create --network host --mount 'type=bind,source=${path}/
passwordfile.txt,target=/opt/payara/passwords/passwordfile.txt,readonly=true'
-e PAYARA_DAS_HOST=${PAYARA_DAS_HOST}  -e PAYARA_CONFIG_NAME=Gruppy-config -e
PAYARA_DEPLOYMENT_GROUP=Gruppy1 cdl-demo:latest
```

This command creates a container from the image we built earlier, using the mount parameter to place our password file into the expected location within the container, and specifying the environment variables used for determining the hostname/IP address of the DAS that we want to contact and the names of the deployment group and config that we want the instance to join and use.

For this example, we've specified the Docker container network type as *host* for simplicity – the Payara Server DAS and Instances expect to be able to talk to each other on the addresses and ports specified in their respective configs.

Starting the Container

When we start the container, an instance on a temporary node with an autogenerated name should be created and started using our config. This instance will join the deployment group, and deploy the application targeted to it, making it available at our configured port of 27070.

You can start the container with the following command, replacing *${containername}* with the ID returned by the previous create command.

```
docker container start ${containername}
```

We can test this by, after waiting a few seconds for the instance to start and deploy the application, hitting the following URL (replacing *${ipaddress}* with your hostname or IP address) to display a simple "Hello World" page:

http://${ipaddress}:27070/rest-jcache

If you go to the *Instances* page on to DAS Admin Console, you should also see the new instance displayed.

Storing Information in the Cache

As noted earlier, the simple application used for this example can store information in JCache by sending data to a JAX-RS endpoint. Use a REST client or the curl command to store some data in there, which we will see replicated to any new instance that joins the deployment group and has the application deployed to it (replace *${ipaddress* with your hostname or IP address):

```
curl -X PUT -H 'Accept: application/json' -H 'Content-Type: application/json'
-i 'http://${ipaddress}:27070/rest-jcache/webresources/cache?key=testy' --data
'{"testy":"westy"}'
```

We can check that this has been stored, by performing a GET request on it like so, from which we should see the data we stored in the cache returned:

http://${ipaddress}:27070/rest-jcache/webresources/cache?key=testy

Creating a Managed Docker Node and Instance

A managed Docker Node is similar to the traditional SSH or CONFIG nodes of Payara Server, where instead of creating the nodes and instances from Docker or other orchestrator, you define the machine from Payara and control Docker via the REST API from Payara using `asadmin` commands.

Creating a Docker Node

A Docker Node can be made using the following `asadmin` command, replacing *${path}* with the absolute path to the password file **on the remote instance** (the machine we're creating our Docker containers on) and *${ipaddress}* with the hostname or IP address:

```
asadmin create-node-docker --nodehost ${ipaddress} -dockerpasswordfile
${path}/passwordfile.txt --dockerimage cdl-demo:latest --dockerport 2376
Docky1
```

This registers the remote machine that the Docker containers are being created on to the DAS from which you can create instances on.

Creating and Starting a Managed Docker Instance

Now that we have a node we can create the instances on, we can create a new one, autogenerate a name for it, add it to our deployment group, and have it use our config with the following command:

```
asadmin --autoname create-instance --node Docky1 --config Gruppy-config
--deploymentgroup Gruppy1
```

This will have Payara talk to the Docker REST API to create a container using our defined image, which we can then start with the standard start-instance `asadmin` command:

```
asadmin start-instance ${instancename}
```

This command will both start the Docker container, and the instance within it.

Once the instance has started, we can check that the application has been deployed to it, and that the value we stored in the cache has been replicated, by hitting the following URL:

http://${ipaddress}:27071/rest-jcache/webresources/cache?key=testy

Viewing Monitoring Data

With Payara Server 5.194 we've integrated a monitoring console, from which instances added via either of the above methods will dynamically appear and disappear as they're added or removed from the domain.

This can be enabled with the following `asadmin` command:

```
asadmin set-monitoring-console-configuration --enabled true
```

This deploys the monitoring console web application to the DAS, which can be found at the following URL (replacing *${ipaddress}* with the hostname or IP address of the DAS):

https://${ipaddress}:8181/monitoring-console

An example screenshot can be seen below:

Managing Unmanaged and Managed Docker Instances

Unmanaged instances, those created by Docker directly, only persist in the domain as long as they are running. Stopping the instance from the DAS will see it immediately deleted, whereas stopping the instance via Docker directly will currently see it deleted upon restart of the DAS. In both cases, the docker container will remain until cleared away or restarted.

Managed instances, those we've created on a Docker Node from Payara Server itself, behave similarly to instances created on SSH or CONFIG nodes, with the Docker containers mirroring the lifecycle of the Payara instance: stopping the instance stops the container, starting the instance starts the container, and deleting the instance will delete the container.

Still have questions? Take a look at all of our container and Docker resources on our website or download Docker images for the Payara Platform here.

Related Guides

Using Payara Server with Docker

Clustering Payara Server in Docker

Using Payara Platform with Docker on Microsoft Azure

sales@payara.fish **+44 207 754 0481** **www.payara.fish**