Hewlett Packard
Enterprise

# Codar/Cloud Service Automation

Software version: 1.90/4.90

For Microsoft Windows® and Linux operating systems

# Docker UCP Datacenter

## Docker Universal Control Plane (UCP)

Document release date: September 2017

Software release date: September 2017

# Legal notices

## Warranty

## Restricted rights legend

## Copyright notice

## Trademark notices

## Documentation updates

To check for recent updates or to verify that you are using the most recent edition of a document, go to the following URL and sign-in or register: https://softwaresupport.hpe.com.

Select Manuals from the Dashboard menu to view all available documentation. Use the search and filter functions to find documentation, whitepapers, and other information sources.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your Hewlett Packard Enterprise sales representative for details.

## Support

Visit the Hewlett Packard Enterprise Software Support Online web site at https://softwaresupport.hpe.com.

# Contents

## Introduction

The containerization technology is rapidly emerging and inspiring many software companies to adopt and use it in their DevOps tool chain to get maximum resource utilization and reduce time-to-market. Docker containerization has become habitual than a buzzword. Docker Universal Control Plane (UCP) offers container datacenter for the application and operations team to install or deploy software; and application with pre-baked dependencies inside a container ensures the application deployment certainty. This whitepaper describes how HPE Cloud Service Automation (CSA) and HPE Codar are integrated with Docker UCP and the benefits the end-user will get by using this integration. HPE CSA is primarily used by the IT Ops or Central IT for providing IaaS & PaaS for their LOB's or application team. HPE Codar offers release pipeline automation along with deployment automation intelligence. Both the products can utilize Docker UCP platform to deploy containers which have software or application in them.

## Configuration Requirements

The following configurations must be completed, tested and should be operational before you proceed with the integration:

- Configure HPE Codar.
- Configure Docker Universal Control Pane
- Configure HPE Operations Orchestration.

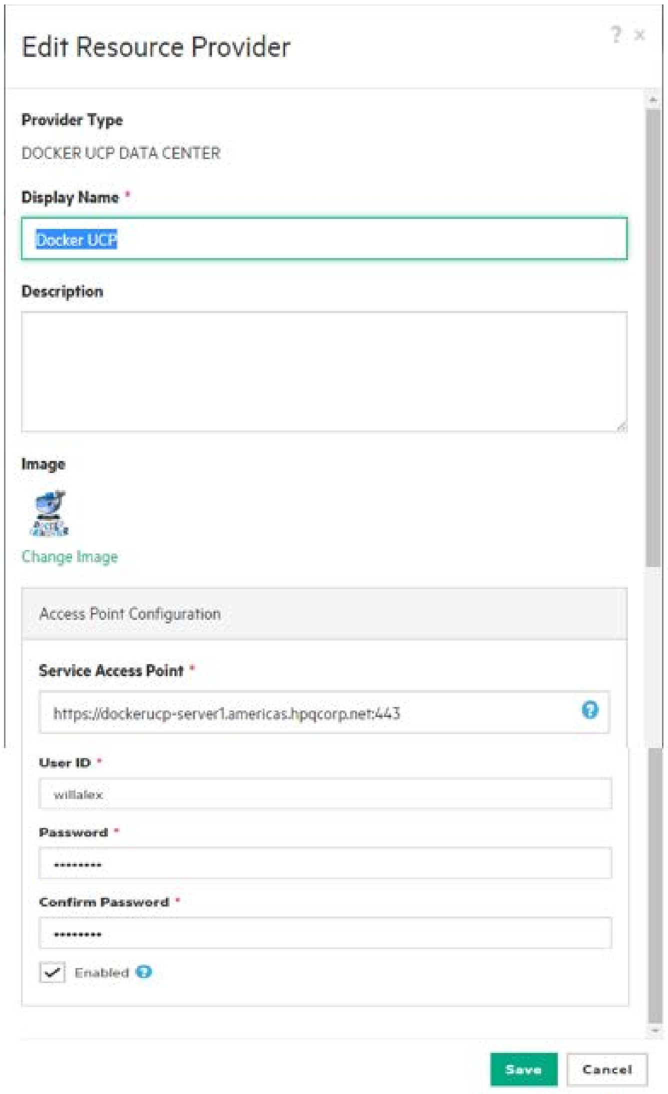
### Supported versions

The following table shows the major components required to use this implementation.

| Component | Supported version | Recommended version |
|---|---|---|
| HPE Codar | 1.81 | 1.81 |
| HPE Operations Orchestration | 10.22 and later | 10.60 with the following HPE Operations Orchestration:<br>• oo10-base-cp-1.8.0.jar<br>• oo10-cloud-cp-1.8.2.jar<br>• oo10-hpe-solutions-cp-1.8.2.jar<br>• oo10-sa-cp-1.2.2.jar<br>• oo10-sm-cp-1.0.3.jar<br>• oo10-virtualization-cp-1.8.0.jar |
| Docker Universal Control Pane | 1.11 | 1.11 |

## Configure Docker UCP Datacenter Resource Provider

Docker UCP Datacenter resource provider can be used to integrate with Docker UCP 1.11 datacenter to import image as component and spawn containers across multiple nodes configured as part of UCP. Docker UCP endpoints with credentials and Docker Trusted Registry endpoints with credentials can be configured within HPE CSA and HPE Codar through OOTB resource provider.


Edit Resource Provider

Provider Type

DOCKER UCP DATA CENTER

Display Name *

Docker UCP

Description

Image

Change Image

Access Point Configuration

Service Access Point *

https://dockerucp-server1.americas.hpqcorp.net:443

User ID *

willalex

Password *

••••••••

Confirm Password *

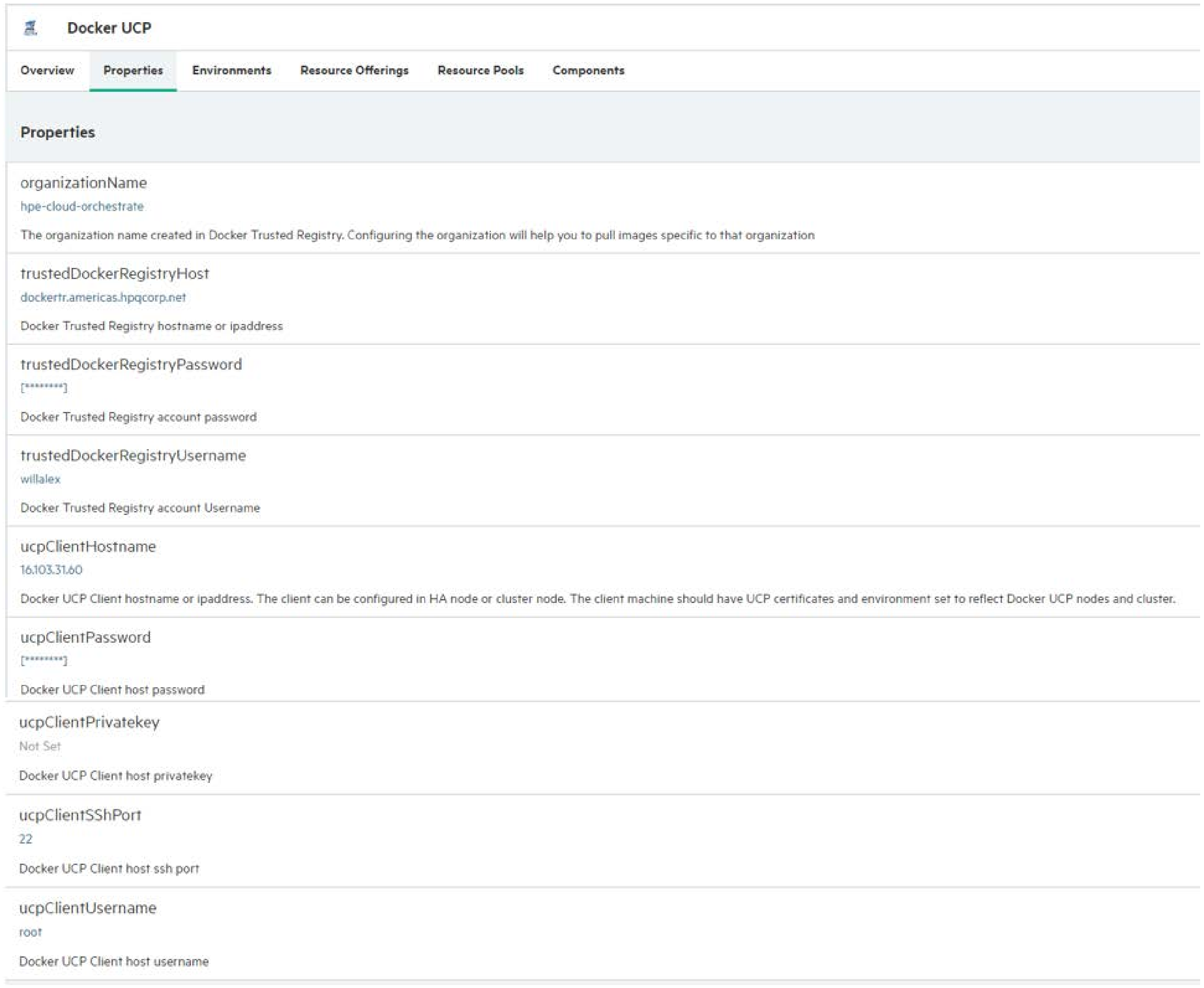
••••••••

✓ Enabled

Instructions:
Provide the Docker UCP controller endpoint and credentials to connect to the UCP server.

Service Access Point: Docker UCP HTTPS endpoint.
User ID: Docker UCP username
Password: Docker UCP password

Fig: 1 → Docker Universal Datacenter Resource Provider


Docker UCP

Overview | Properties | Environments | Resource Offerings | Resource Pools | Components

Properties

organizationName
hpe-cloud-orchestrate
The organization name created in Docker Trusted Registry. Configuring the organization will help you to pull images specific to that organization

trustedDockerRegistryHost
dockertr.americas.hpqcorp.net
Docker Trusted Registry hostname or ipaddress

trustedDockerRegistryPassword
[********]
Docker Trusted Registry account password

trustedDockerRegistryUsername
willalex
Docker Trusted Registry account Username

ucpClientHostname
16.103.31.60
Docker UCP Client hostname or ipaddress. The client can be configured in HA node or cluster node. The client machine should have UCP certificates and environment set to reflect Docker UCP nodes and cluster.

ucpClientPassword
[********]
Docker UCP Client host password

ucpClientPrivatekey
Not Set
Docker UCP Client host privatekey

ucpClientSShPort
22
Docker UCP Client host ssh port

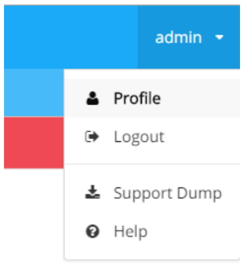
ucpClientUsername
root
Docker UCP Client host username

Fig: 2 → Additional properties to configure “Docker Trusted Registry” endpoints and credentials.

## UCP Client

A UCP client can be used to connect to Docker UCP controller node and send request to create or terminate containers.

The UCP client machine should be a UNIX machine and should be configured with UCP client bundle. The client bundle can be downloaded from Docker UCP server. Ensure to add the env.sh variables into user’s home **“.bashrc”, “.profile” and “/etc/environment”** files. The UCP client machine can also be one of Docker UCP’s HA nodes, cluster nodes, or Docker containers. Refer detailed description for each of the properties in Fig 2.



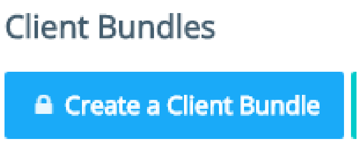


Fig: 3 → Download the client bundle files from the Docker UCP controller portal

The client bundles can be download using command line option and also by executing the below commands on the client machine:

```
cd /home/<username>
mkdir ucpcerts
AUTHTOKEN=$(curl -sk -d '{"username":"<username>","password":"<password>"}' https://<controller>/auth/login | jq -r .auth_token)
curl -k -H "Authorization: Bearer $AUTHTOKEN" https://<controller>/api/clientbundle -o bundle.zip
```

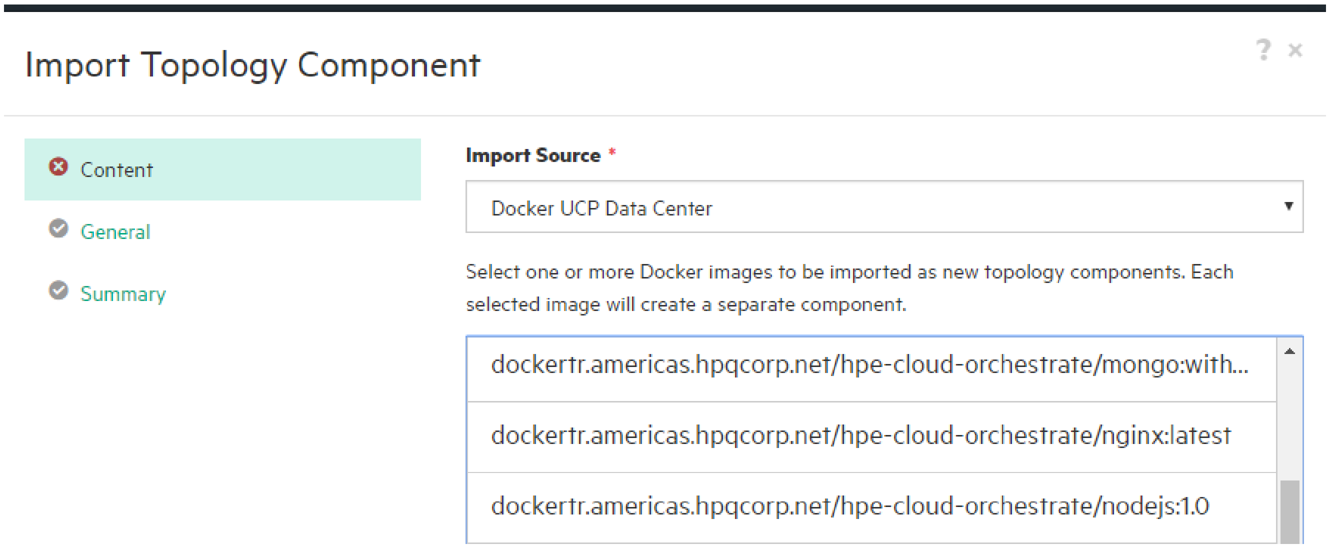
```
Add the environment variables into user's home directory .bashrc , .profile and /etc/environments

export NO_PROXY=localhost,127.0.0.0/8,::1,/var/run/docker.sock,<additional>
export DOCKER_HOST=tcp://<UCP Controller FQDN/IP Address>:443
export DOCKER_TLS_VERIFY=1
export DOCKER_CERT_PATH=</home/<username>/ucpcerts
```

# Embrace Docker Image as Component

HPE CSA and HPE Codar support topology-based designs. The components are the building blocks for the service blueprints and can be imported from various resources like HPE Operations Orchestration, HPE Server Automation, Chef, Puppet, etc.

In the same fashion, a Docker image can be imported from Docker-trusted registry/registry as component, and can be added as part of the service blueprints. A Docker image once imported can be further configured to set the dependency over other Docker components which attributes to another Docker image or any other component from any other resource provider.

The images can be imported based on the organization configured in the DOCKER UCP DATA CENTER resource provider.



Import Source *

Docker UCP Data Center

Select one or more Docker images to be imported as new topology components. Each selected image will create a separate component.

dockertr.americas.hpqcorp.net/hpe-cloud-orchestrate/mongo:with...

dockertr.americas.hpqcorp.net/hpe-cloud-orchestrate/nginx:latest

dockertr.americas.hpqcorp.net/hpe-cloud-orchestrate/nodejs:1.0

Fig: 4 → Import Docker images as components by selecting one or more from the list

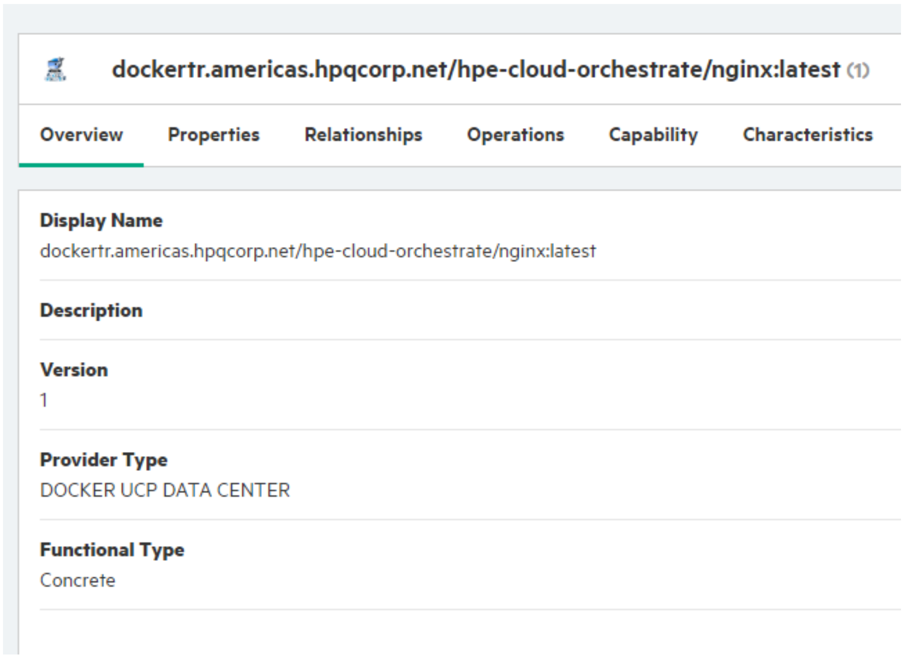
**Imported Components**

The imported component needs to be configured for the relationship, if required, and for setting the capabilities and characteristics. The default operations available are Deploy and Undeploy. Other custom operations can be imported from the content.

The custom operation includes the following:

1. Start
2. Stop

3. Restart
4. Inspect
5. Get Logs
6. GetStats


dockertr.americas.hpqcorp.net/hpe-cloud-orchestrate/nginx:latest (1)

Display Name
dockertr.americas.hpqcorp.net/hpe-cloud-orchestrate/nginx:latest

Description

Version
1

Provider Type
DOCKER UCP DATA CENTER

Functional Type
Concrete

Fig: 5 → A sample Docker UCP component

## Component Properties

The component properties include input, output, and provider properties. The input properties are given below.

**Note**: For information on passing user inputs to individual “docker run” command line arguments, refer the section ’Docker ’run‘ command Line Arguments as Component Properties’.

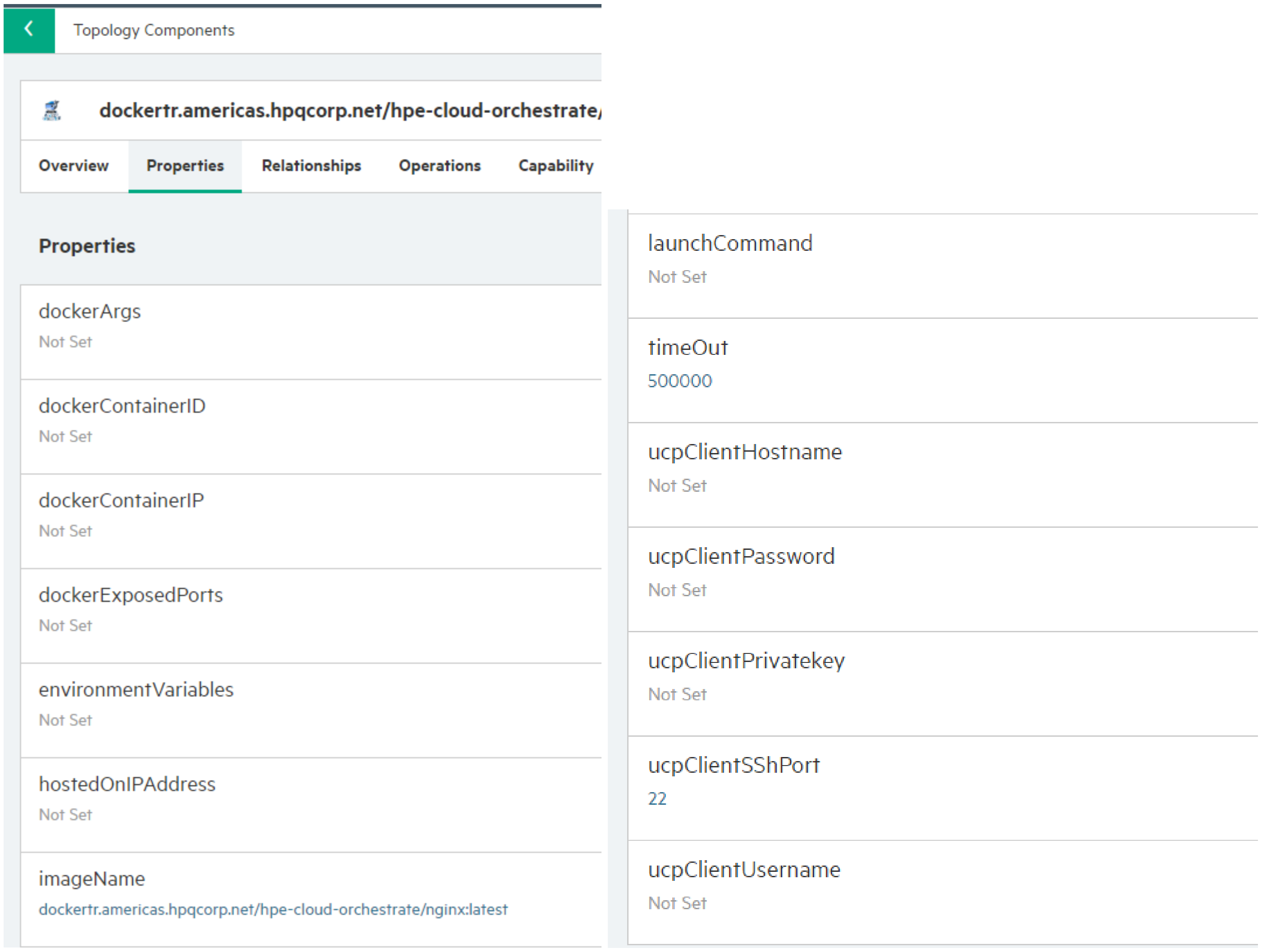
*Refer the information below to pass inputs to create containers in a non-controlled environment. If it is made visible to consumer, the end-users can pass the docker run options on their own and pass any inputs based on their requirement. They can be hardcoded too and these arguments need not be made visible.*

| **Property** | **Description** |
|---|---|
| dockerArgs | All the Docker “run” arguments can be provided which can create containers with specified options.<br>Example: -P / -p 18080:8080 –v /tomcat-8:/usr/local/tomcat8 |
| environmentVariables | Optional argument but environment variables can be provided as part of dockerArgs also.<br>Ex: -e http_proxy=http://mydomainproxy.mydomain.com:8080/<br>-e artifactURL=http://mynexus/com/product/myweb.war |
| launchCommand | An entrypoint or launch command which should be triggered inside container once it is created.<br>Eamplex: /run.sh |

You can mask component properties like ’dockerContainerIDfrom end-user by unchecking the options like ’designer visible and consumer visible ’.

**Note:** Modifying the image name will not incur any change and it is strongly recommended not to modify the default “imageName” property.

All the other property values are automatically fetched from the provider property.



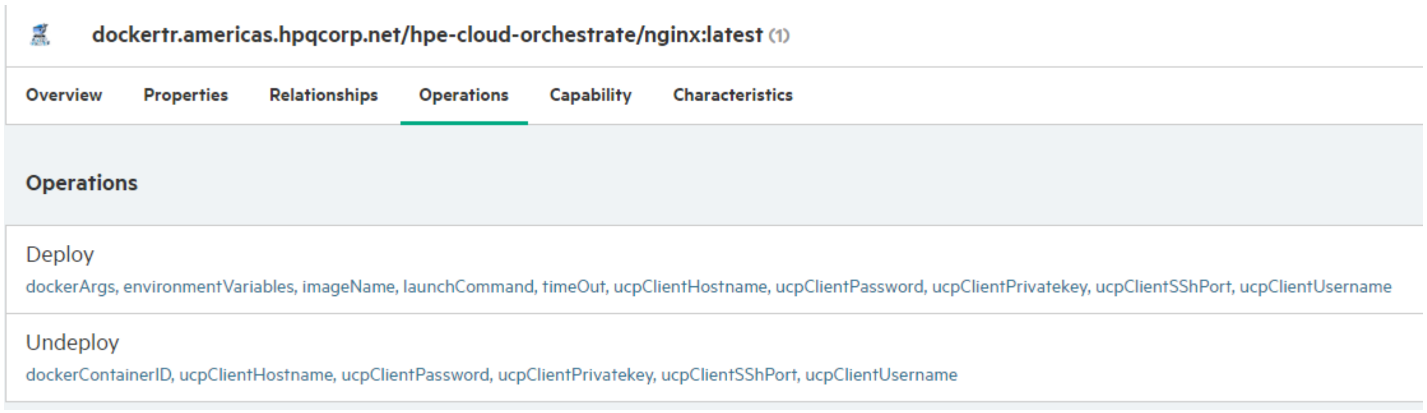
Topology Components

dockertr.americas.hpqcorp.net/hpe-cloud-orchestrate/

**Properties**

| | |
|---|---|
| dockerArgs | Not Set |
| dockerContainerID | Not Set |
| dockerContainerIP | Not Set |
| dockerExposedPorts | Not Set |
| environmentVariables | Not Set |
| hostedOnIPAddress | Not Set |
| imageName | dockertr.americas.hpqcorp.net/hpe-cloud-orchestrate/nginx:latest |
| launchCommand | Not Set |
| timeOut | 500000 |
| ucpClientHostname | Not Set |
| ucpClientPassword | Not Set |
| ucpClientPrivatekey | Not Set |
| ucpClientSShPort | 22 |
| ucpClientUsername | Not Set |

Fig: 6 → Default Component Properties



dockertr.americas.hpqcorp.net/hpe-cloud-orchestrate/nginx:latest (1)

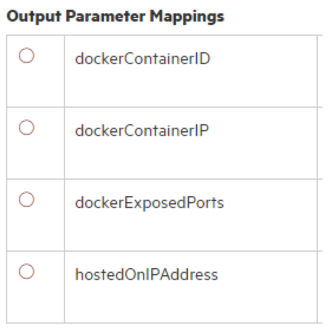
**Operations**

| | |
|---|---|
| Deploy | dockerArgs, environmentVariables, imageName, launchCommand, timeOut, ucpClientHostname, ucpClientPassword, ucpClientPrivatekey, ucpClientSShPort, ucpClientUsername |
| Undeploy | dockerContainerID, ucpClientHostname, ucpClientPassword, ucpClientPrivatekey, ucpClientSShPort, ucpClientUsername |

Fig: 7 → Default operations as part of Docker UCP component

The Deploy operation will have some properties exposed as output properties which can be used further in different components. There is no need to change these property mappings.

The properties include the following:



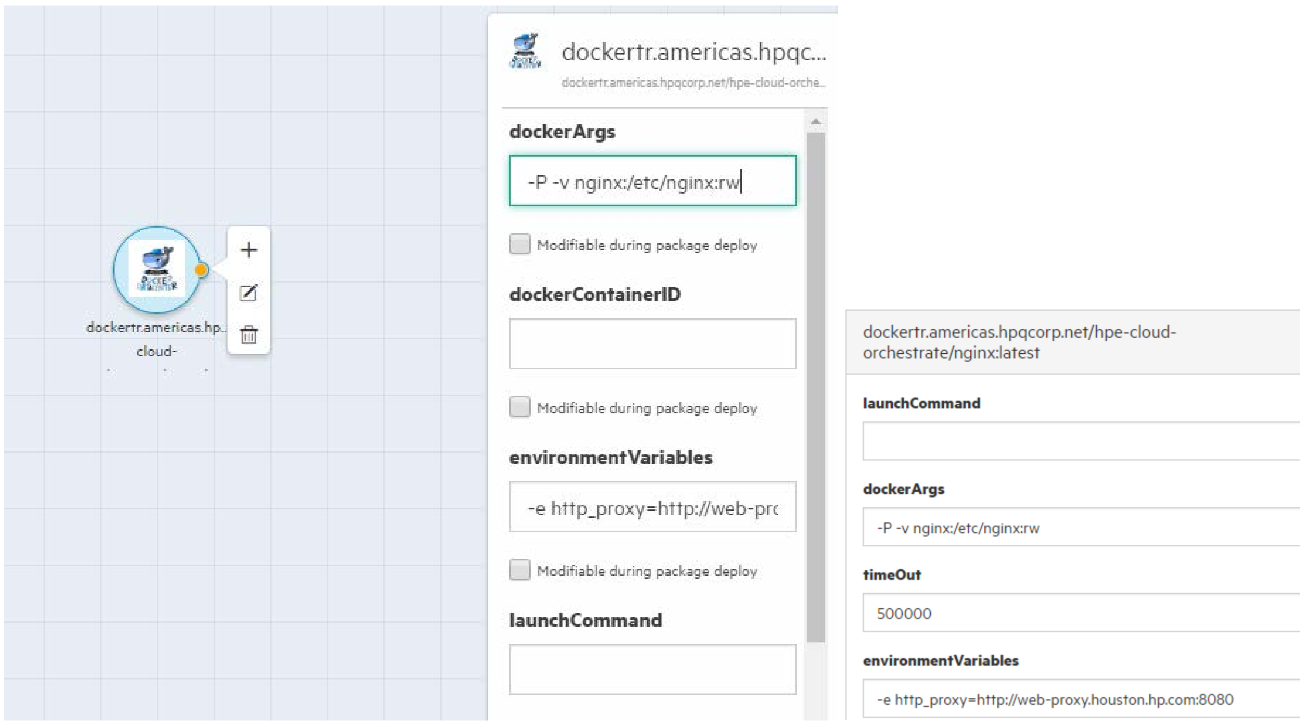
**Output Parameter Mappings**

| | |
|---|---|
| ○ | dockerContainerID |
| ○ | dockerContainerIP |
| ○ | dockerExposedPorts |
| ○ | hostedOnIPAddress |

Fig: 8 → Deploy flow output properties

# Application Blueprints

You can create application blueprints by adding the imported components. A sample application blueprint is given below with a component which represents an "nginx" Docker image.


dockertr.americas.hp...
cloud-
dockertr.americas.hpqc...
dockertr.americas.hpqcorp.net/hpe-cloud-orche...

dockerArgs

-P -v nginx:/etc/nginx:rw

Modifiable during package deploy

dockerContainerID

Modifiable during package deploy

environmentVariables

-e http_proxy=http://web-prc

Modifiable during package deploy

launchCommand

dockertr.americas.hpqcorp.net/hpe-cloud-orchestrate/nginx:latest

**launchCommand**

**dockerArgs**

-P -v nginx:/etc/nginx:rw

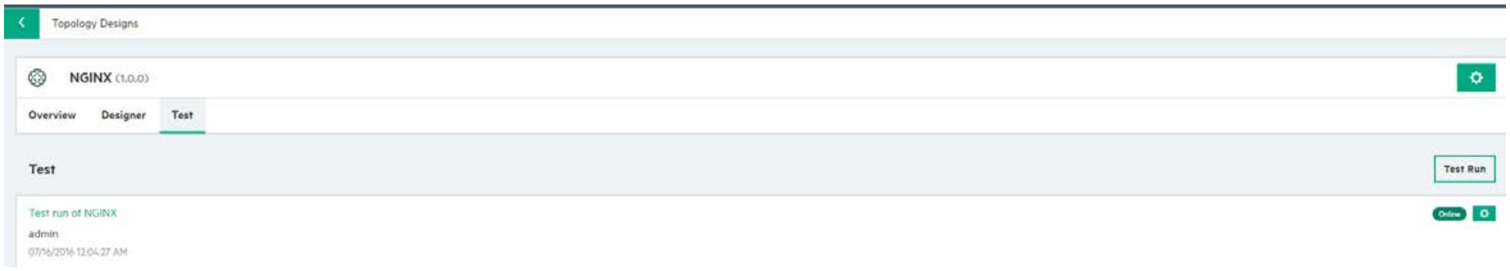
**timeOut**

500000

**environmentVariables**

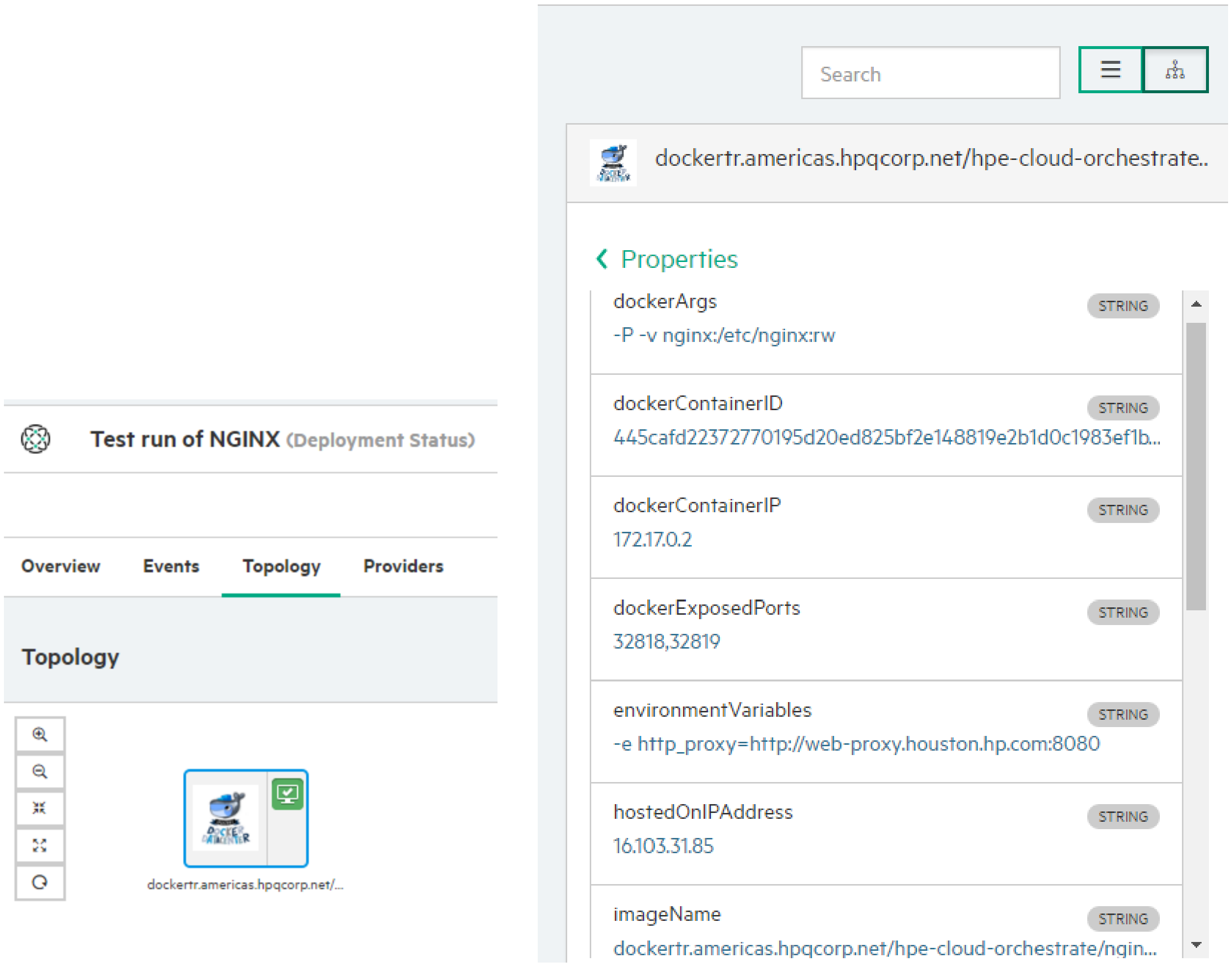
-e http_proxy=http://web-proxy.houston.hp.com:8080

Fig: 9 → Sample Docker image component which represents nginx server



Test run of NGINX
admin
07/16/2016 12:04:27 AM

Fig 10: Test Run Result


Test run of NGINX (Deployment Status)
Overview
Events
Topology
Providers
Topology
dockertr.americas.hpqcorp.net/...
Search
dockertr.americas.hpqcorp.net/hpe-cloud-orchestrate..
Properties

| | | |
|---|---|---|
| dockerArgs | -P -v nginx:/etc/nginx:rw | STRING |
| dockerContainerID | 445cafd22372770195d20ed825bf2e148819e2b1d0c1983ef1b... | STRING |
| dockerContainerIP | 172.17.0.2 | STRING |
| dockerExposedPorts | 32818,32819 | STRING |
| environmentVariables | -e http_proxy=http://web-proxy.houston.hp.com:8080 | STRING |
| hostedOnIPAddress | 16.103.31.85 | STRING |
| imageName | dockertr.americas.hpqcorp.net/hpe-cloud-orchestrate/ngin... | STRING |

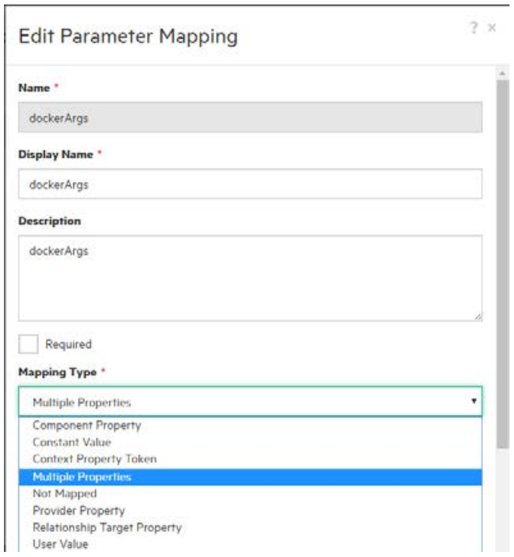
Fig 11 → Topology view with output properties like docker container ID, published ports, and IP address it is hosted on.

## Docker "run" Command Line Arguments as Component Properties

This section describes how you can pass an end-user's input or an input from external system as part of Docker run arguments which include ports, volumes, environment variables, etc. This can help the end-user to avoid creating "docker-compose" yml file; and using the topology design, one can completely stand up Docker service with all the necessary actions like publishing the ports, linking to another container, creating volumes, etc.

## Special Component Property “dockerArgs”

A topology component property can take inputs from various types through property mapping types as given below,



Edit Parameter Mapping

Name *
dockerArgs

Display Name *
dockerArgs

Description
dockerArgs

☐ Required

Mapping Type *
Multiple Properties

- Component Property
- Constant Value
- Context Property Token
- Multiple Properties
- Not Mapped
- Provider Property
- Relationship Target Property
- User Value

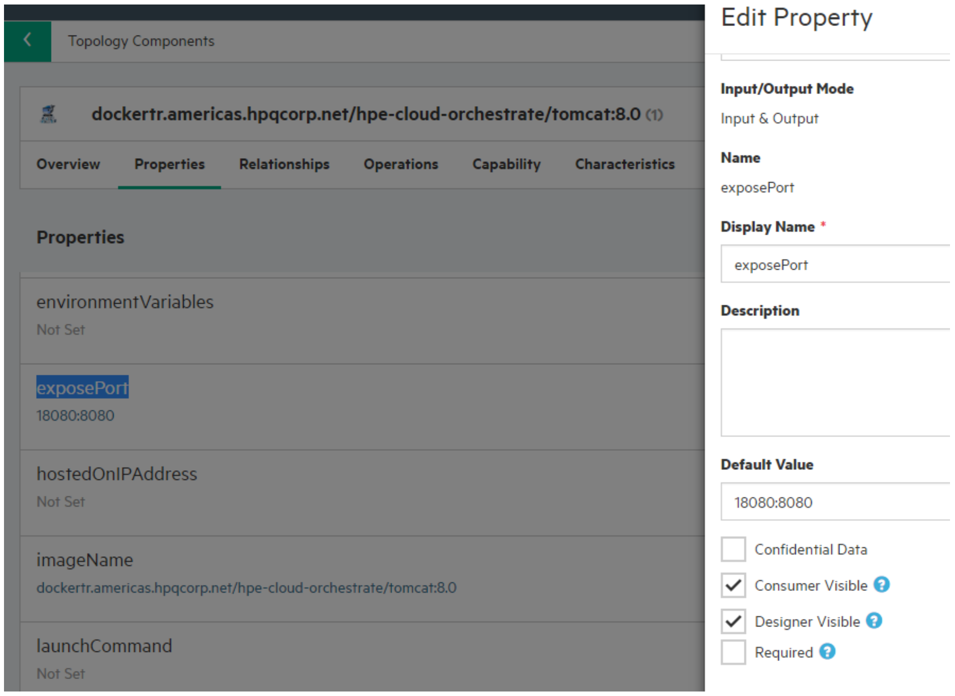
Fig:12 → List of mapping types

This “Docker UCP Datacenter” content capsule has the capability to take inputs from various mapping types as given below:

For example,

If an application team wants to receive the ports to be published from end-user, below steps can be performed,

1. Create a component property named “exposePort” and enable the options Designer and Consumer visible



Topology Components

dockertr.americas.hpqcorp.net/hpe-cloud-orchestrate/tomcat:8.0 (1)

Overview | Properties | Relationships | Operations | Capability | Characteristics

Properties

| | |
|---|---|
| environmentVariables | Not Set |
| exposePort | 18080:8080 |
| hostedOnIPAddress | Not Set |
| imageName | dockertr.americas.hpqcorp.net/hpe-cloud-orchestrate/tomcat:8.0 |
| launchCommand | Not Set |

Edit Property

Input/Output Mode
Input & Output

Name
exposePort

Display Name *
exposePort

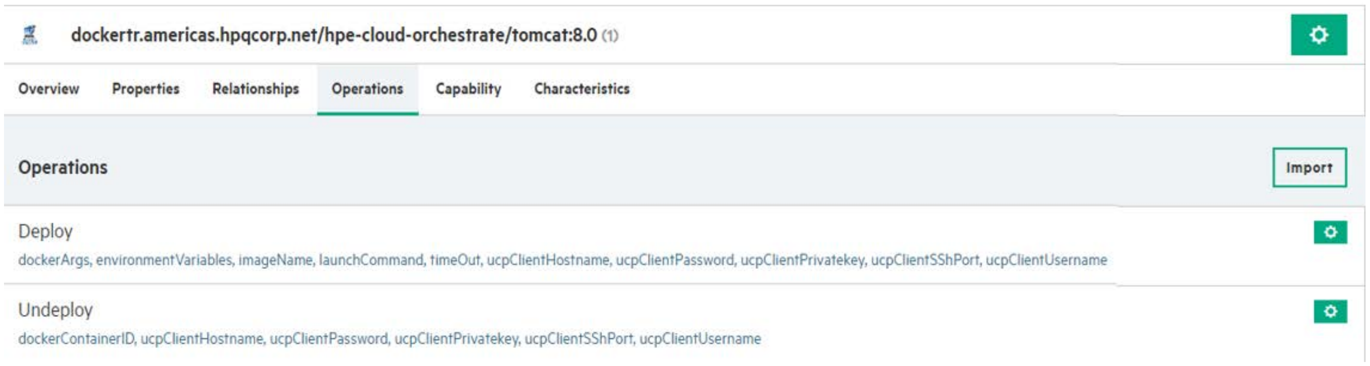
Description

Default Value
18080:8080

☐ Confidential Data
✓ Consumer Visible
✓ Designer Visible
☐ Required

Fig: 13 → Create component property

2. Go to “Operations” →Deploy → Edit (gear ⚙ ).

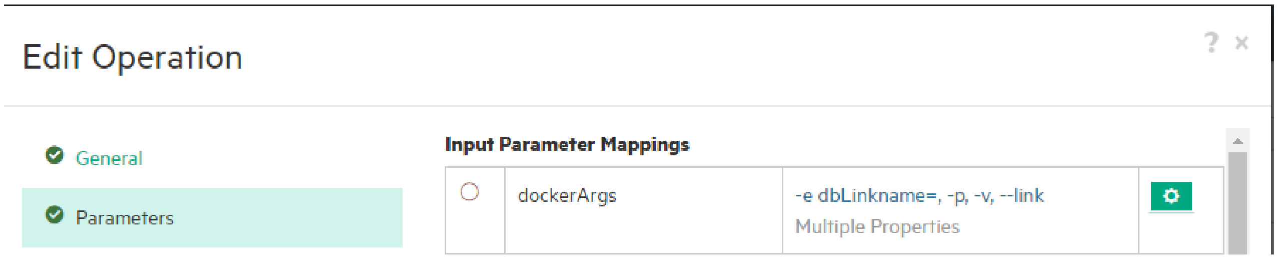

dockertr.americas.hpqcorp.net/hpe-cloud-orchestrate/tomcat:8.0 (1)

Operations

Deploy
dockerArgs, environmentVariables, imageName, launchCommand, timeOut, ucpClientHostname, ucpClientPassword, ucpClientPrivatekey, ucpClientSShPort, ucpClientUsername

Undeploy
dockerContainerID, ucpClientHostname, ucpClientPassword, ucpClientPrivatekey, ucpClientSShPort, ucpClientUsername

3. Click Parameters.

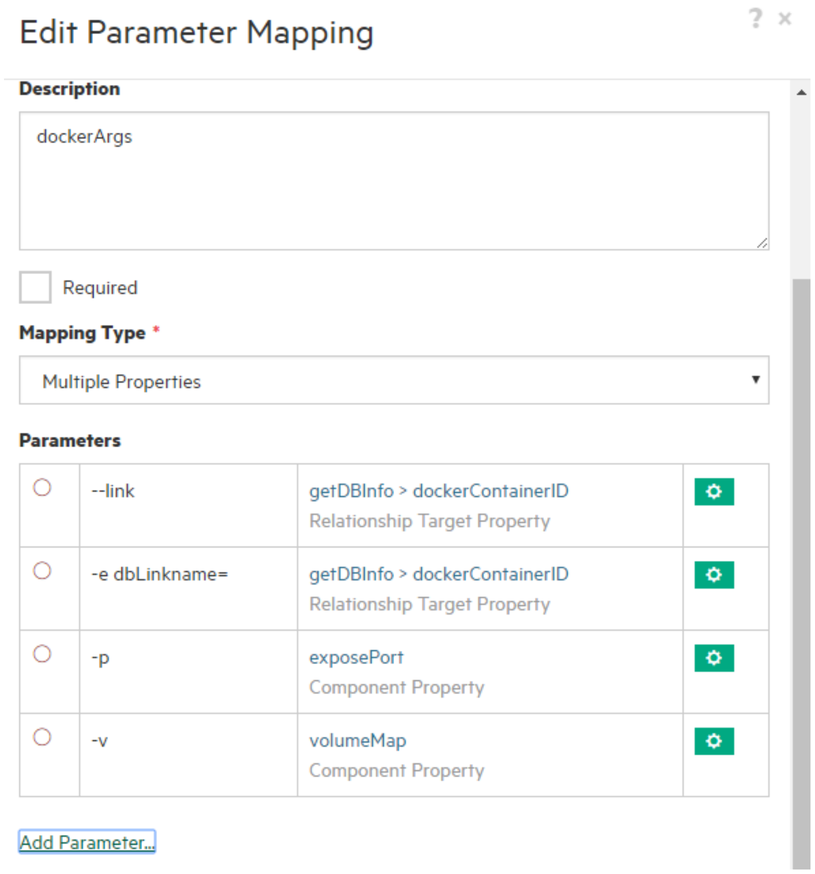

Edit Operation

Input Parameter Mappings

| | | | |
|---|---|---|---|
| ○ | dockerArgs | -e dbLinkname=, -p, -v, --link<br>Multiple Properties | ⚙ |

Fig :14 → Special Docker UCP component property "dockerArgs"

4. Click "dockerArgs" gear box ⚙ and edit the property.


Edit Parameter Mapping

Description
dockerArgs

☐ Required

Mapping Type *
Multiple Properties

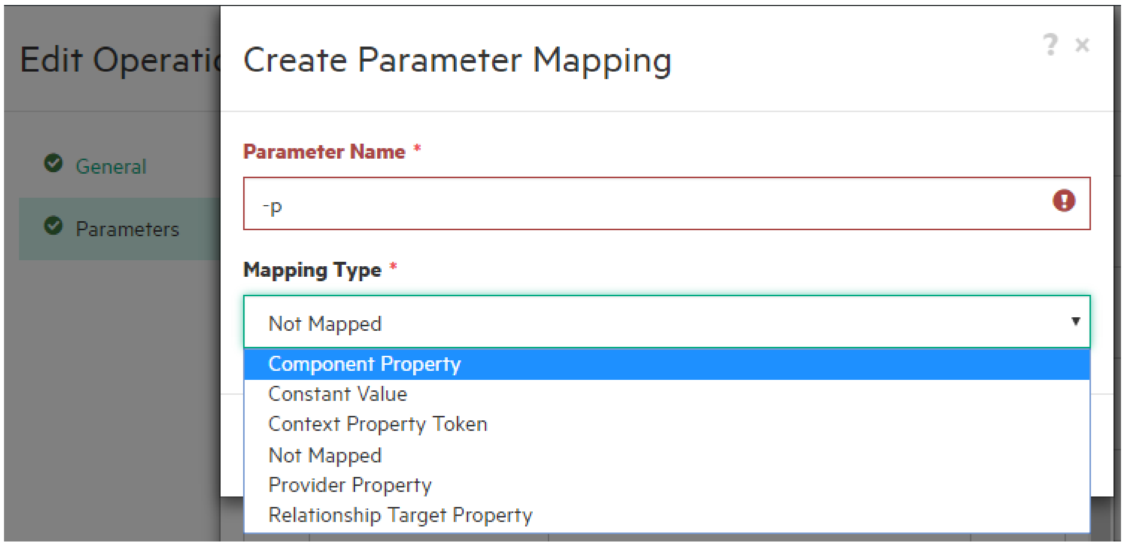
Parameters

| | | | |
|---|---|---|---|
| ○ | --link | getDBInfo > dockerContainerID<br>Relationship Target Property | ⚙ |
| ○ | -e dbLinkname= | getDBInfo > dockerContainerID<br>Relationship Target Property | ⚙ |
| ○ | -p | exposePort<br>Component Property | ⚙ |
| ○ | -v | volumeMap<br>Component Property | ⚙ |

Fig: 15 → Edit parameter mapping to add additional parameter

5. Select 'Multiple Properties' as the mapping type option.

6. Now start adding the parameter using 'Add Parameter' link.



**Parameter Name** *

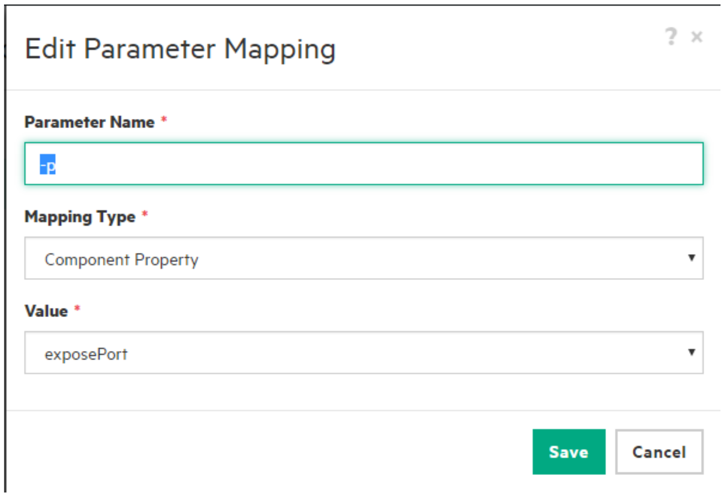
-p

**Mapping Type** *

Not Mapped

- Component Property
- Constant Value
- Context Property Token
- Not Mapped
- Provider Property
- Relationship Target Property

Fig: 16-1 → List of mapping type to pass the values to parameter



**Parameter Name** *

-p

**Mapping Type** *

Component Property

**Value** *

exposePort

Fig: 16-2: Component mapping type selected to pass input from user to “-p” option to publish ports (example)

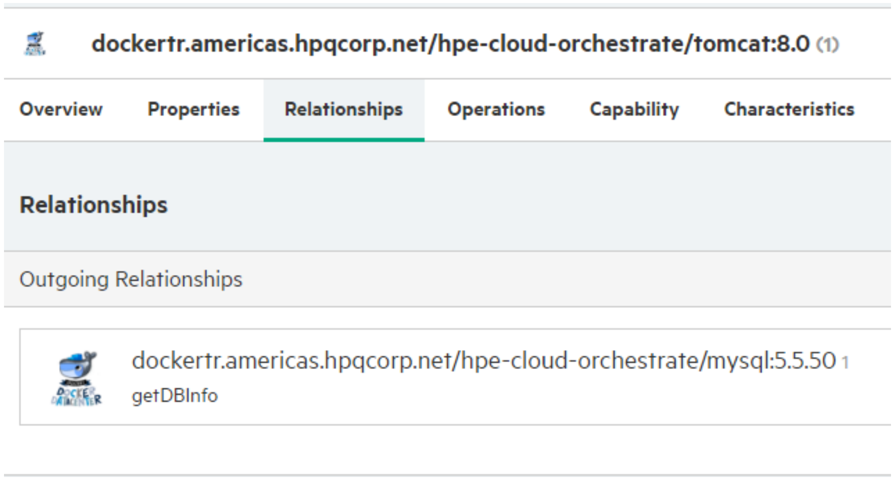
7. Select the option accordingly from where the inputs should be passed to this parameter. For example, to get input from end-user, select the ’Component Property‘ mapping type,
8. To get an input from ’Relationship Target Property‘, select the mapping type accordingly. For example, if there is a dependency over another Docker component, native HPE Operation Orchestration component, or any other topology component, this approach can be used.

## Container Linking

The special component property ’dockerArgs” facilitates container linking. It helps you to use the Docker linking feature

**—‘--link ‘** by setting a relationship to another Docker UCP component or Docker component using container ID which is exposed as the output parameter. After the deployment, the container ID will get generated and will be passed as input to the Docker UCP component which is set as relationship.

In the example below, the Tomcat 8.0 Docker UCP has a relationship with MySQL Docker UCP component and using this parameter mapping, the end-user can pass —‘--link ‘ Docker run option as the parameter name and the value can come from the relationship property set for to get the MySQL Docker UCP Component.


dockertr.americas.hpqcorp.net/hpe-cloud-orchestrate/tomcat:8.0 (1)

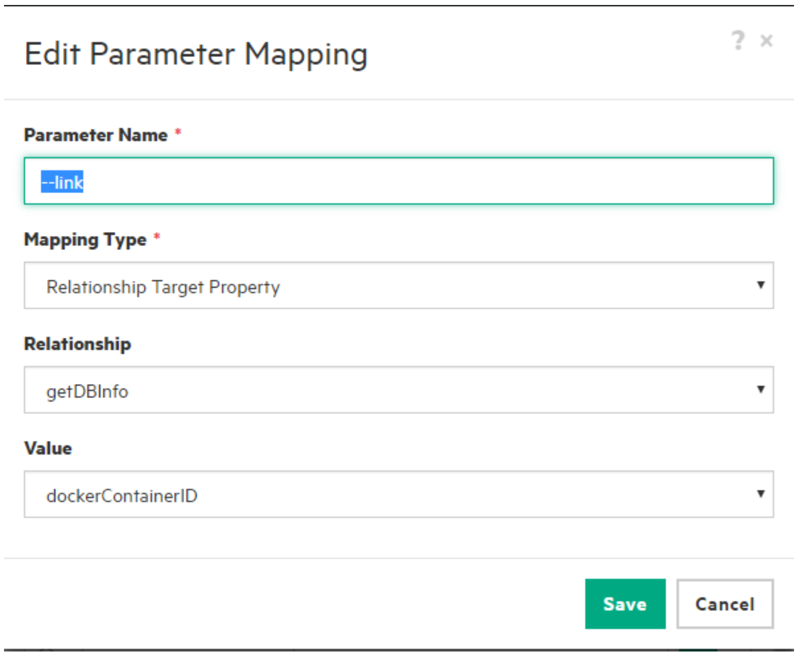
Overview | Properties | Relationships | Operations | Capability | Characteristics

Relationships

Outgoing Relationships

dockertr.americas.hpqcorp.net/hpe-cloud-orchestrate/mysql:5.5.50 1
getDBInfo

Fig: 17 → Sample Docker UCP component with dependency set on another Docker UCP component using topology "Relationships"


Edit Parameter Mapping

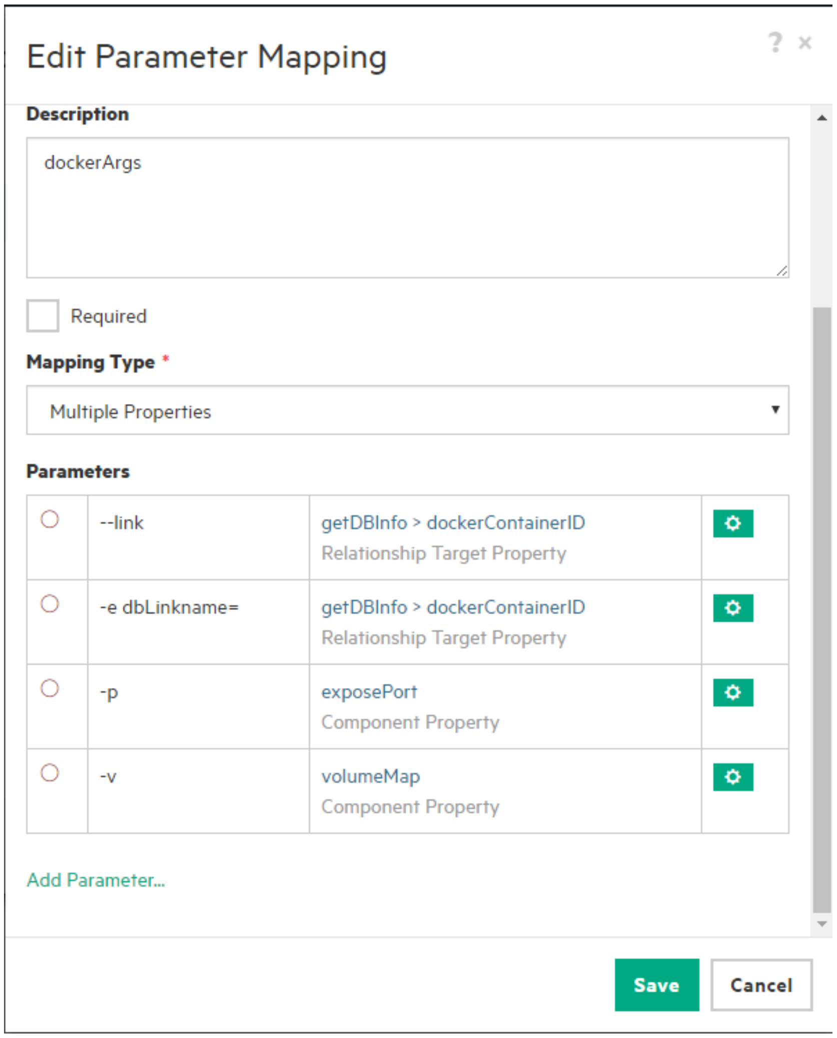
Parameter Name *: --link

Mapping Type *: Relationship Target Property

Relationship: getDBInfo

Value: dockerContainerID

Fig: 18 → Way to provide link option as part of "docker run" command and pass the value fetched from another container realized through Docker UCP component.



Edit Parameter Mapping

Description

dockerArgs

Required

Mapping Type *

Multiple Properties

Parameters

| | | |
|---|---|---|
| --link | getDBInfo > dockerContainerID<br>Relationship Target Property | |
| -e dbLinkname= | getDBInfo > dockerContainerID<br>Relationship Target Property | |
| -p | exposePort<br>Component Property | |
| -v | volumeMap<br>Component Property | |

Fig: 19 → List of sample possible inputs to pass to the dockerArgs special component property.

## How to "link" another container using relationship

In figure 19, multiple inputs are provided to dockerArgs. For example, ports to be published, volume to be created, linking to another container through container id and to get the link name pass the container id information as environment name.

The parameter '**-e dbLinkname='** will automatically create an environment name on the container. This environment variable can be used to identify the respective dependent container environment variables which are set on the component's Relationships tab as part of topology component.

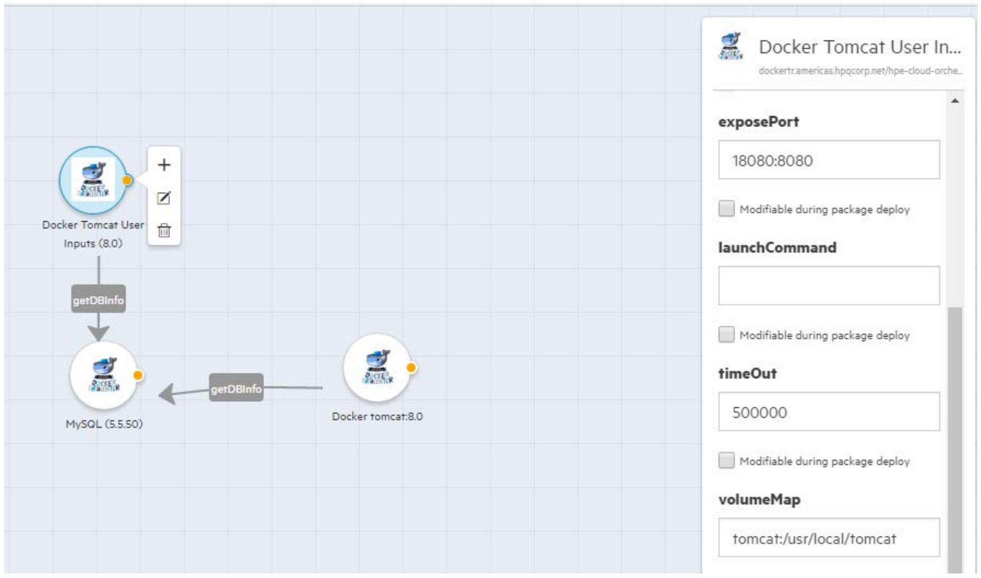
**Note:** the symbol '=' should be provided in order to pass any environment variables with '-e' option.

Syntax: '**-e <environment name>=** ' (Set the appropriate mapping type)

## Component Property "environmentVariables"

When using the environment variables in the dockerArgs, it is optional to use the input component property "environmentVariables" but the component will still accept any inputs passed as part of this input property.

## Sample Designs



Docker Tomcat User In...

dockertr.americas.hpqcorp.net/hpe-cloud-orche...

exposePort: 18080:8080

Modifiable during package deploy

launchCommand:

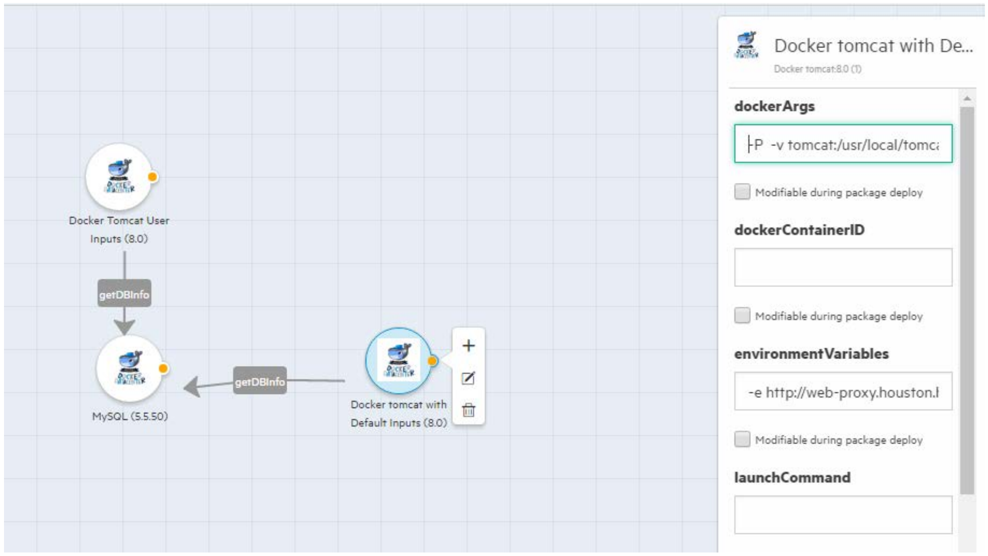
Modifiable during package deploy

timeOut: 500000

Modifiable during package deploy

volumeMap: tomcat:/usr/local/tomcat

Fig: 20 → A sample design which take ports to publish and volume to map as input component property


Docker Tomcat User Inputs (8.0)
getDBInfo
MySQL (5.5.50)
getDBInfo
Docker tomcat with Default Inputs (8.0)

Docker tomcat with De...

Docker tomcat:8.0 (1)

dockerArgs: -P -v tomcat:/usr/local/tomc

Modifiable during package deploy

dockerContainerID:

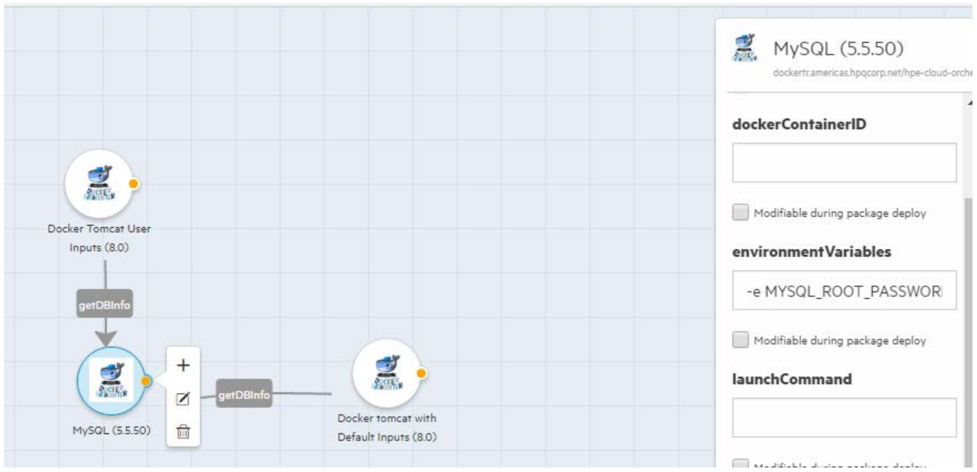
Modifiable during package deploy

environmentVariables: -e http://web-proxy.houston.l

Modifiable during package deploy

launchCommand:

Fig: 21 → A sample design which take inputs using default input component property


Docker Tomcat User Inputs (8.0)
getDBInfo
MySQL (5.5.50)
getDBInfo
Docker tomcat with Default Inputs (8.0)

MySQL (5.5.50)

dockertr.americas.hpqcorp.net/hpe-cloud-orche

dockerContainerID:

Modifiable during package deploy

environmentVariables: -e MYSQL_ROOT_PASSWORI

Modifiable during package deploy

launchCommand:

Fig: 22 → A sample design which has the link to different components

## Send documentation feedback

If you have comments about this document, you can send them to clouddocs@hpe.com.