Liqid Kubernetes Integration Utility

v3.4

Copyright (c) 2024 by Liqid, Inc. - All Rights Reserved

Table of Contents

[Liqid and Kubernetes 3](#__RefHeading___Toc142_4286456116)

[Support for Multi-Tenancy 4](#__RefHeading___Toc144_4286456116)

[Principles of Operation 5](#__RefHeading___Toc169_4286456116)

[Typical Workflow 6](#__RefHeading___Toc169_4286456116_Copy_1)

[Usage 7](#__RefHeading___Toc98_4286456116)

[Invoking the Utility 7](#__RefHeading___Toc123_4286456116)

[Logging (-l switch) 7](#__RefHeading___Toc125_4286456116)

[Ignoring Warnings (-f switch) 8](#__RefHeading___Toc127_4286456116)

[Dry-run execution (-no switch) 8](#__RefHeading___Toc129_4286456116)

[Common Switches 8](#__RefHeading___Toc134_4286456116)

[-ip={ip\_address} 8](#__RefHeading___Toc136_4286456116)

[-u={username} 8](#__RefHeading___Toc148_4286456116)

[-p={password} 9](#__RefHeading___Toc150_4286456116)

[-px={url} 9](#__RefHeading___Toc138_4286456116)

[Command Reference 10](#__RefHeading___Toc100_4286456116)

[Adopt 10](#__RefHeading___Toc102_4286456116)

[Annotate 11](#__RefHeading___Toc104_4286456116)

[Machine annotations 11](#__RefHeading___Toc159_4286456116)

[Resource annotations 11](#__RefHeading___Toc161_4286456116)

[Automatic Annotation 13](#__RefHeading___Toc163_4286456116)

[Clearing Annotations 14](#__RefHeading___Toc165_4286456116)

[Compose 15](#__RefHeading___Toc106_4286456116)

[Initialize 16](#__RefHeading___Toc108_4286456116)

[Link 18](#__RefHeading___Toc110_4286456116)

[Nodes 19](#__RefHeading___Toc112_4286456116)

[Release 20](#__RefHeading___Toc114_4286456116)

[Reset 21](#__RefHeading___Toc116_4286456116)

[Resources 22](#__RefHeading___Toc118_4286456116)

[Unlink 23](#__RefHeading___Toc120_4286456116)

# Liqid and Kubernetes

TODO

# Support for Multi-Tenancy

TODO

# Principles of Operation

TODO

# Typical Workflow

TODO

# Usage

## Invoking the Utility

The integration utility is implemented as a Java program and is provided as a standard .jar file, which may be executed by any recent JRE. Liqid recommends openjdk version 20.

The utility may be invoked by typing the following:

java -jar k8s-integration.jar {switches} {command}

A select number of commands have been implemented, and each command accepts and in some cases requires a particular set of switches. The entire selection of switches and commands can be obtained by typing:

java -jar k8s-integration.jar --help

The particular version of the utility may be obtained by typing:

java -jar k8s-integration.jar --version

Command Line Switches

Command line switches generally follow the conventions established for unix utilities. They may be specified in their long or short forms, and those switches which accept argument values may be specified either as:

-switch=value

or as:

-switch value

A few switches will accept multiple values, and can be specified as:

-switch=value1,value2,...

or as:

-switch value1 -switch value2 ...

## Logging (-l switch)

Certain diagnostic information may be written to stdout and/or to stderr. More specific diagnostics can be enabled by adding the -l switch, which enables verbose logging. Such logging will be placed in a file named "liq.config.log."

During trouble-shooting, Liqid support personnel may request that the -l switch is enabled, and that the resuling log file is sent to Liqid for analysis.

## Ignoring Warnings (-f switch)

While preparing to carry out user requests, the utility may detect certain speficiation or configuration anomalies which could produce unexpected results. When this happens, an error message is displayed and the utility stops before taking any actions. It may be the case that several messages are displayed, but *generally*, no action will be taken when such anomalies are detected.

This behavior can be overridden with the -f switch, which forces action even in the presence of such anomalies. This switch is to be used with caution.

## Dry-run execution (-no switch)

Most commands accept the -no option, which prevents the utility from taking any action. The utility \*will\* analyze command line input as well as Liqid and Kubernetes Cluster configuration states, it \*will\* create a plan for altering the Cluster states, but it will stop before actually executing the plan.

## Common Switches

Many comands share a certain subset of switches. These are listed below:

### -ip={ip\_address}

The -ip switch accepts an argument representing the IP address or DNS name for the Liqid Cluster Director. This is configured at install time for the Liqid Cluster, and is necessary for this utility to communicate with the Liqid Cluster to check the configuration of the Cluster, and to effect any necessary configuration changes.

### -u={username}

The -u switch specifies a username credential which will be used by this utility when communicating with the Liqid Cluster. It is expected that the Cluster is configured to use either no authentication or local username/password authentication. LDAP and OpenID are not currently supported.

If the Liqid and Kubernetes Clusters are linked with credentials, then the credentials stored with Kubernetes are normally used for accessing the Liqid Cluster. Specifying this switch will override such behavior.

### -p={password}

The -p switch specifies a password credential which will be used by this utility when communicating with the Liqid Cluster. It is expected that the Cluster is configured to use either no authentication or local username/password authentication. LDAP and OpenID are not currently supported.

If the Liqid and Kubernetes Clusters are linked with credentials, then the credentials stored with Kubernetes are normally used for accessing the Liqid Cluster. Specifying this switch will override such behavior.

### -px={url}

The -px switch accepts an argument representing the URL for a kubectl proxy. Information regarding the specification and use of kubectl proxy is available elsewhere, but generally it exists to allow simplified access to the Kubernetes API. This utility leverages the kubectl proxy facility for communication with Kubernetes.

Ensure that the proxy is configured such that it will accept all API requests from the host on which this utility is running.

Be aware that the use of kubectl proxy carries certain security concerns. Ensure that you are taking proper steps to secure your Kubernetes Cluster when using kubectl proxy.

Example invocation for the kubectl proxy:

kubectl proxy --port=8080

If the kubectl proxy is running on the same host as is the utility, then the -px switch might look like:

-px=http://127.0.0.1:8080

# Command Reference

This section describes the various implemented commands, providing information on how, why, and when to use each command.

## Adopt

This command is used when new resources are added to the Liqid Cluster, and are to be made available to this utility. It is presumed that a Liqid and Kubernetes cluster has already been established and is being managed via annotations and by this utility.

For compute nodes, the following actions are taken:

* A Liqid Machine is created for the compute node, and is given a name which reflects both the resource name (formatted as "pcpu{n}") and the Kubernetes worker name
* The Liqid compute node resource will be linked with Kubernetes via a user description in the Liqid resource, which contains the name of the Kubernetes worker node
* The Kubernetes worker will be linked with the Liqid Machine via an annotation on the Kubernetes worker node which contains the Liqid Machine name

The compute node is expected to already be participating as a worker node in the Kubernetes cluster, with all relevant software and plug-ins installed, including (but not limited to) kubelet and kube-proxy.

For other resource nodes such as GPUs or FPGAs, the following action is taken:

* The resource is moved into the group free pool for the Liqid Group which is associated with the Kubernetes Cluster.

The resource is now eligible for automatic composition via the compose command.

Syntax:

adopt

-px,--proxy-url={proxy\_url}

[ -pr,--processors={pcpu\_name:worker\_node\_name}[,...] ]

[ -r,--resources={name}[,...] ]

[ -f,--force ]

[ -no,--no-update ]

For example, if you have added a compute node identified as "pcpu5" and it is known to Kubernetes as "worker-5", you would type something like:

java -jar k8s-integration.jar -px=http://127.0.0.1:8080 -pr=pcpu5:worker-5

If you added several GPU nodes identified as "gpu7", "gpu8", and "gpu9", you might type:

java -jar k8s-integration.jar -px=http://127.0.0.1:8080 -r=gpu7,gpu8,gpu9

## Annotate

This command is used to indicate, for a particular Kubernetes worker node, the Liqid Resource specifications which apply to that node. It is provided as a convenient alternative to requiring the adminsitrator to apply the annotations manually via kubectl.

The Compose command relies upon the various worker node annotations for working out which devices should be composed into which machines.

### Machine annotations

There is one machine annotation per worker node. It is formatted as:

kubint.liqid.com/machine-name={machine\_name}

This annotation ties the worker node to the Liqid Machine which contains the compute node associated with the worker node itself. The Liqid Machine will contain one compute resource, and potentially many other resources such as GPUs and FPGAs.

### Resource annotations

A resource specification consists of a list of resource model specifications which describe, in aggregate, the general profile of the resources which should be composed into the machine containing the given worker node.

A resource specification is made up of one, two, or three parts, and applies to a specific resource type (FPGA, GPU, MEMORY, LINK, or SSD).

A Generic Resource Specification consists of only a count, representing the number of such resources from any vendor, and of any model, which should be composed into the containing machine. It is an integer formatted simply as:

{count}

A Vendor Resource Specification consists of a vendor name and a count, indicating the number of such resources from that vendor, which should be composed into the containing machine. It is formatted as:

{vendor\_name}:{count}

A Specific Resource Specification consists of a vendor name, model, and count, indicating the number of such resources from that vendor and of that model, which should be composed into the containing machine. It is formatted as:

{vendor\_name}:{model\_name}:{count}

Resource specifications are generally additive. That is, if one specifies

ACME:T4:2,ACME:3,2

meaning a count of 2 ACME T4 GPUs, 3 more ACME GPUs of any model, and finally 2 more GPUs of any vendor and model, the total specification is for 2 + 3 + 2 = 7 GPUs. They might all be ACME T4s, but there will certainly be at least two of T4s as well as three more (for a total of five) ACME GPUs, and potentially as many as 7, although the last two most general GPUs could also be from INITECH or INITRODE.

One may also specify alternative vendors and models, for example:

INITECH:IH-4:3,INITRODE:ID-1000:2,3

indicating 3 INITECH IH-4 FPGAs, 2 INITRODE ID-1000 FPGAs, and 3 FPGAs of any vendor.

There is a special case where a count of zero can be used in conjunction with other specifications.

A Specific Resource Specification of zero for a particular vendor and model indicates that no resources of the indicated type matching that vendor and model should be composed into the machine. This could be presented in order to allow all models from a vendor except for a particular model. In other words, one could specify:

ACME:5,ACME:T500:0

which means 5 SSDs from ACME, but none of the ACME T500 models. In this case, 5 SSDs would be composed into the machine of any model from ACME except for T500s.

A Vendor Resource Specification of zero indicates that no resources of the indicated type from that vendor should be composed into the machine. For example:

12,INITECH:0

indicating that twelve GPUs of any vendor and model should be composed into the machine, excepting that no GPUs from INITECH should be allowed.

Finally, a Generic Resource Specification of zero indicates that no resources of the indicated type should be composed into the machine at all. This special case does not exist in the actual worker node annotations; rather, it is specified on the command line to indicate that any existing annotation of that type should be removed. This would prevent any resources of that type from being composed into the machine.

These types are given as annotations on the relevant worker node formatted as:

kubint.liqid.com/fpga-resources={fpga\_spec},...

kubint.liqid.com/gpu-resources={gpu\_spec},...

kubint.liqid.com/link-resources={link\_spec},...

kubint.liqid.com/memory-resources={mem\_spec},...

kubint.liqid.com/ssd-resources={ssd\_spec},...

A complete specification be annotated as:

kubint.liqid.com/gpu-resources=ACME:T200:5,ACME:2

kubint.liqid.com/link-resources=NETZ:T25:2,NETZ:10,5

and it could be effected by the following:

java -jar k8s-integration.jar -px=http:127.0.0.1:8080 -n=worker5 -m=pcpu5-worker5 \

-gs=ACME:T200:5,ACME:2 \

-ls=NETZ:T25:2,NETZ:10,5 \

-fs=0

Syntax:

annotate

-px,--proxy-url={proxy\_url}

-n,--worker-node={worker\_node\_name}

[ -m,--liqid-machine={liqid\_machine} ]

[ -fs,--fpga-spec={spec}[,...] ]

[ -gs,--gpu-spec={spec}[,...] ]

[ -ls,--link-spec={spec}[,...] ]

[ -ms,--mem-spec={spec}[,...] ]

[ -ss,--ssd-spec={spec}[,...] ]

[ -f,--force ]

[ -no,--no-update ]

### Automatic Annotation

A special case of this command exists which evaluates the inventory of resources assigned to the relevant Liqid Cluster Group or to machines within that group, allocating them as evenly as possible across all of the properly-labeled worker nodes.

For example, if there are a total of 15 GPUs and 8 FPGAs, and there are 5 worker nodes in the configuration, the automatic algorithm would assign 3 GPUs to each of the worker nodes and either 1 or 2 FPGAs to each of the worker nodes.

No resources will actually be moved; no composition takes place. Rather, annotations are simply rewritten to reflect the calculated allocations.

Syntax:

annotate

-px,--proxy-url={proxy\_url}

-a,--automatic

[ -f,--force ]

[ -no,--no-update ]

### Clearing Annotations

A special case of this command exists which is designed to clear all annotations from a particular worker node. This includes annotations for all resource specifications. It does NOT clear the machine annotation.

Syntax:

annotate

-px,--proxy-url={proxy\_url}

-n,--worker-node={worker\_node\_name}

-cl,--clear

[ -f,--force ]

[ -no,--no-update ]

## Compose

This command evaluates the difference between the current composition of the Liqid Machines and the desired composition as defined by the Kubernetes worker node annotations, develops a plan for composing the various machines in as efficient and least-disruptive manner as possible, then executes that plan.

The algorithm for determining which reconfigurations are necessary is complex, but it is designed to allow a typical Kubernetes configuration to continue operating with little or no production downtime.

Specifically, adding resources to a machine has little effect on the corresponding worker node. However, removing resources requires that the worker node becomes unschedulable, and all pods on the machine must be evicted. Once the resource(s) is/are removed, the worker node is made schedulable, and Kubernetes will eventually bring up pods on that machine again.

This is done due to the potential deleterious effects on any particular pod resulting from suddently having a required resource taken away from it.

It is suggested that this command be executed first with the -no switch, so that the adminstrator can examine the proposed plan before allowing the utility to go ahead with the compose function.

Syntax:

compose

-px,--proxy-url={proxy\_url}

[ -f,--force ]

[ -no,--no-update ]

## Initialize

This command creates the linkage and annotations necessary for eventual resource composition management. The following conditions should exist prior to invoking the command:

* The Kubernetes deployment is complete and functioning properly
* The Liqid Cluster compute nodes which are intended to participate in the Kubernetes deployment as worker nodes, are in fact properly deployed as workers.
* The Liqid Cluster compute nodes are currently not composed into machines, nor are they assigned to any Liqid Cluster group free pool.
* The Liqid Cluster resources (such as FPGAs and GPUs) which are to participate in the Kubernetes deployment are not currently composed into any machines, nor are they assigned to any Liqid Cluster group free pool.
* The Liqid Cluster group which is intended to hold the various resources and compute nodes has not yet been created.

Certain of these requirements can be waived if you use the -f switch. For example, the resources will be removed from machines and/or groups, the identifier group will be removed, and other certain actions will be taken to ensure a clean configuration prior to initialization. None of these additional actions will be taken in the absence of the -f switch.

When this command is invoked, the following major actions are undertaken:

* Any conflicting configuration state will be resolved, but only if the -f switch is specified
* A linkage will be created which stores Liqid Cluster information and (optionally) Liqid credentials in the Kubernetes database.
* A group will be created to contain the various machines which will be composed for the Kubernetes deployment
* Liqid machines will be composed within the Liqid Cluster group, per compute node
* Liqid Cluster compute nodes will have descriptions applied to them, linking the node to the corresponding Kubernetes worker name
* Kubernetes nodes will be annotated to refer to the Liqid Cluster machines containing the corresponding compute nodes
* The called-out resources (such as FPGAs and GPUs) will be assigned to the Liqid Cluster group free pool

Additionally, the -al switch may be specified which causes the following additional actions to be taken:

The worker nodes will be annotated with resource specifications which allocate resources across the worker nodes, as evenly as possible

Those resources will be composed into the corresponding Liqid Cluster machines, such that they become available for use by the various pods in the corresponding worker nodes

Any clients of the Liqid Cluster apart from the Kubernetes Cluster in question, will not be affected. Specifically, any resources or compute nodes which are not specified for this command, will be ignored.

Specifying compute nodes

The compute nodes are specified as a list of arguments to the -pr switch. Each specification is presented as a colon-seperated pair of identifiers:

compute\_node\_name : worker\_node\_name

For example, the following specification:

-pr=pcpu0:worker-0,pcpu1:worker-1,pcpu2:worker-2

would be appropriate for indicating that the Liqid compute node identified as pcpu0 is to be associated with the Kubernetes woker node named worker-0, pcpu1 with worker-1, and pcpu2 with worker-2. Again, it is expected that the compute nodes are already installed and deployed as Kubernetes worker nodes with the indicated Kubernetes node names.

The composable resources are specified as a simple list of resource names. For example, four GPUs and two FPGAs would likely be specified as:

-r=gpu0,gpu1,gpu2,gpu3,fpga0,fpga1

Syntax:

initialize

-px,--proxy-url={proxy\_url}

-ip,--liqid-ip-address={ip\_address}

[ -u,--liqid-username={user\_name} ]

[ -p,--liqid-password={password} ]

-g,--liqid-group={group\_name}

-pr,--processors={pcpu\_name:worker\_node\_name}[,...]

-r,--resources={name}[,...]

[ -al,--allocate ]

[ -f,--force ]

[ -no,--no-update ]

## Link

Before using the Compose command, it is necessary to link the Kubernetes deployment with the Liqid Cluster. This consists mainly of recording a small amount of information regarding how to access the Liqid Cluster Director in the Kubernetes database.

While this could be done manually, the utility provides the Link command for carrying out this process more easily.

When invoked, the command will:

* Record the IP address of the Liqid Cluster Director in the Kubernetes database
* Record the name of the Liqid Cluster group associated with this Kubernetes deployment in the Kubernetes database
* Optionally record credentials for accessing the Liqid Cluster Director as a secret, in the Kubernetes database

Syntax:

link

-px,--proxy-url={proxy\_url}

-ip,--liqid-ip-address={ip\_address}

[ -u,--liqid-username={user\_name} ]

[ -p,--liqid-password={password} ]

-g,--liqid-group={group\_name}

[ -f,--force ]

[ -no,--no-update ]

## Nodes

This command lists all of the worker nodes from the perspective of Kubernetes. Of particular interest would be the pods currently assigned to the node, as well as the Liqid-specific annotations.

The -f and -no switches are not supported as this command makes no configuration changes to either the Liqid Cluster nor to Kubernetes.

Syntax:

nodes

-px,--proxy-url={proxy\_url}

## Release

This command effectively removes knowledge of Liqid Cluster resources from Kubernetes. Use this when compute nodes or other resources are taken out of the Liqid Cluster, or are reassigned for other uses.

Supposing one wishes to stop using two GPUs and a compute node, one might type:

java -jar k8s-integration.jar release -px=http://127.0.0.1:8080 -r=gpu4,gpu5,pcpu3

Syntax:

release

-px,--proxy-url={proxy\_url}

-r,--resources={name}[,...]

[ -f,--force ]

[ -no,--no-update ]

## Reset

Do not use this command.

It will remove all Liqid annotations from all worker nodes.

It will remove Liqid linkage information (Liqid IP address and group name) from the Kubernetes database.

It will completely unconfigure the Liqid Cluster, including any groups and machines which have nothing to do with the Kubernetes deployment.

If you do use this command, use the -no switch first to see the reletively short (but very dangerous) plan which will be created. Then don't use it anyway.

Syntax:

reset

-px,--proxy-url={proxy\_url}

-ip,--liqid-ip-address={ip\_address}

[ -u,--liqid-username={user\_name} ]

[ -p,--liqid-password={password} ]

-f,--force

[ -no,--no-update ]

## Resources

This command lists the Liqid Cluster inventory. Relevant information includes the names, vendors, and models of the various resources, and the current assignation of such resources to the various machines.

The -f and -no switches are not supported as this command makes no configuration changes to either the Liqid Cluster nor to Kubernetes.

Syntax:

resources

-ip,--liqid-ip-address={ip\_address}

[ -u,--liqid-username={user\_name} ]

[ -p,--liqid-password={password} ]

## Unlink

This command reverses the action of the Link command.

Specifically, it removes the Kubernetes ConfigMap entries which contain the Liqid Cluster information for the cluster associated with this Kubernetes deployment, as well as the secret Liqid credentials if they are stored in the Kubernetes database.

This command does not remove annotations from the worker nodes.

Use this command when you no longer intend to use this utility for automatic composition of Liqid resources in conjunction with the Kubernetes deployment, or when you wish to re-establish the information from scratch (although the Link command would do this anyway).

Syntax:

unlink

-px,--proxy-url={proxy\_url}

[ -f,--force ]

[ -no,--no-update ]