Intel® Deep Learning Studio Installation and Configuration

Installation Guide, Beta Release

Intel Confidential
Revision 1.0: December 2018

Document Revision History

Document Revision Number	Date	Comments
1.0	December 2018	Beta Release

Terms and Conditions

This document contains information on products and/or processes in development.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS AND/OR SERVICES. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN AN EXECUTED AGREEMENT, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS AND SERVICES INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN A SIGNED WRITING BY INTEL, THE INTEL PRODUCTS AND SERVICES ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product /process descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products and services described in this document may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Contents

Figures

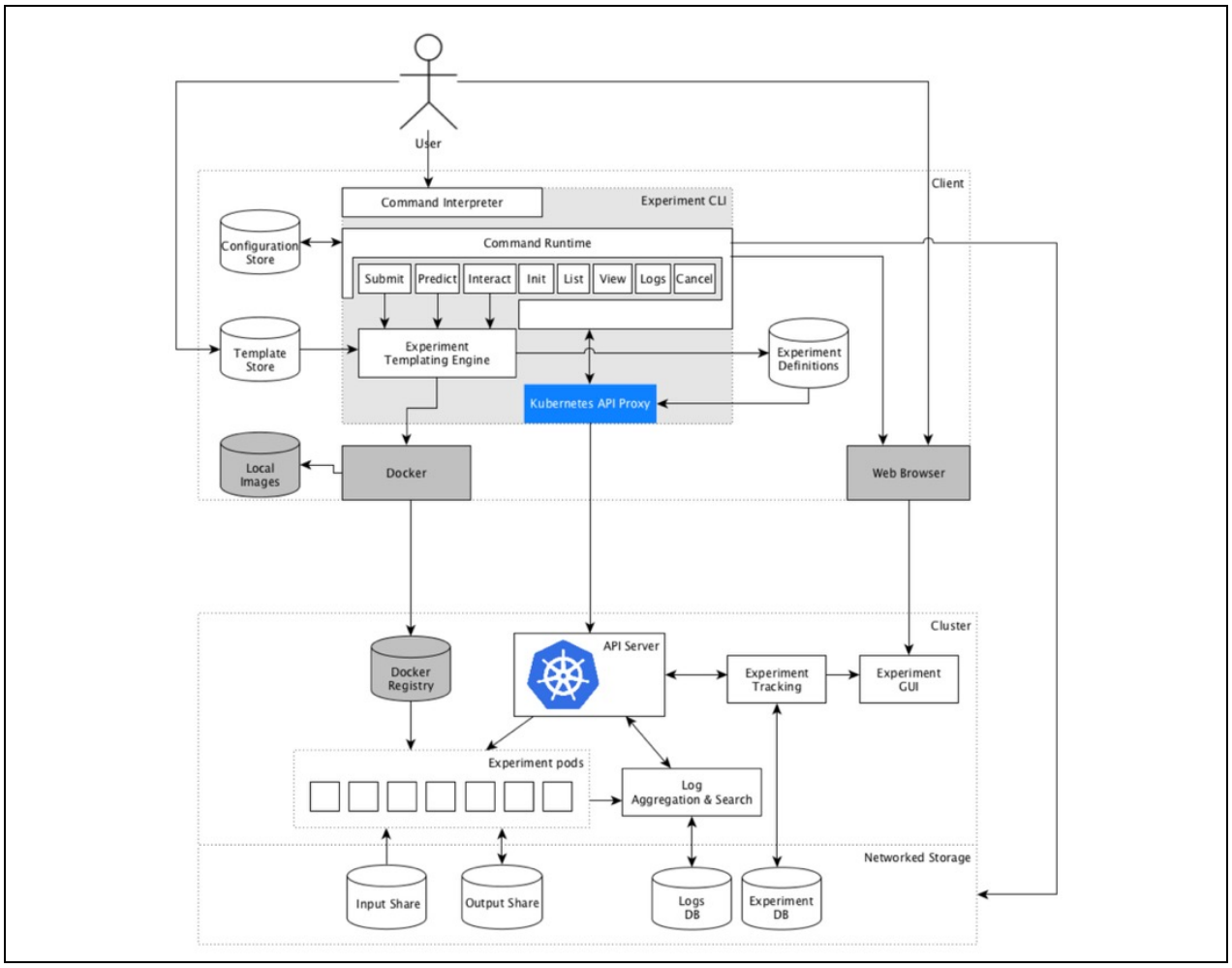
Tables

Introduction to Intel® Deep Learning Studio

The Intel® Deep Learning Studio (Intel® DL Studio) guide provides an overview of the Intel DL Studio requirements, general installation requirements, as well as the installation process, and inventory and configuration tasks to set up a test environment. However, you may have additional networking, configuration management, or operating system requirements, in which case consult with your IT department guidelines.

Figure 1 shows the Intel DL Studio Networking Diagram.

Figure 1: Intel DL Studio Software System Architecture



Intel® DL Studio Components System Requisites

Intel DL Studio requires certain components, as shown in Table 1.

List of Included External Software Components

The Intel DL Studio software package includes the external software components listed in Table 1.

Table 1: Intel DL Studio Included Software Components

Name	Version	Project link
addon-resizer	1.7	http://k8s.gcr.io/addon-resizer
dashboard	1.8.3	https://k8s.gcr.io/kubernetes-dashboard-amd64
defaultbackend	1.4	https://github.com/kubernetes/ingress-nginx
dnsmasq-nanny	1.14.8	https://github.com/kubernetes/dns
dns-sidecar	1.14.8	https://github.com/kubernetes/dns
elasticsearch	6.2.3	https://github.com/elastic/elasticsearch
etcd	3.3.9	https://github.com/coreos/etcd
flannel	0.9.1	https://github.com/coreos/flannel
fluentd	0.12	https://www.fluentd.org
heapster	1.4.2	https://github.com/kubernetes/heapster
ingress	0.14.0	http://quay.io/kubernetes-ingress-controller/nginx-ingress-controller
kubectl	1.10.6	https://github.com/kubernetes/kubernetes/tree/master/pkg/kubectl
kube-dns	1.14.8	https://github.com/kubernetes/dns
kube-proxy	1.10.6	http://gcr.io/google-containers/kube-proxy-amd64
mkl-dnn	0.14	https://github.com/intel/mkl-dnn
nginx	1.14.0	https://www.nginx.com
pause	3.1	http://gcr.io/google-containers/pause-amd64
redsocks	0.5	https://github.com/darkk/redsocks
registry	2.6.2	https://github.com/docker/distribution
tensorflow	1.9.0	https://github.com/tensorflow/tensorflow
tiller	2.9.1	https://github.com/helm/helm

Key Components

Key Components include:

- An Optimized HW stack
- Intel DL Studio complete bundle (including installation scripts and reference configuration).
  - Vanilla Kubernetes cluster on top of provisioned HW and OS-level SW
  - Intel DL Studio components (containerized components, their configuration, integration means, and so on.)
  - Used SW components are optimized for IA containers with Intel optimized libraries such as: Math Kernel Library (Intel® MKL) for Deep Neural Networks (Intel® MKL-DNN), optimized framework parameters, and so on.

Included Documentation

- Installation and Configuration Administration Guide
  - Format = PDF
- User Guide
  - Format = HTML, PDF
- Release Notes
  - Format = PDF

Intel® DL Studio System Requirements

Intel DL Studio supported installer systems, include:

- Red Hat* Enterprise Linux* 7.5
- Ubuntu* 16.04
- CentOS* 7.5

Red Hat Enterprise Linux 7.5

Required on system, software requirements:

- sshpass (when password authentication is used)

CentOS 7.5

Required on system, software requirements:

- sshpass (when password authentication is used)

Ubuntu 16.04

Required on system, software requirements:

- Python* 3.5
- apt required packages:
  - python3-pip
  - build-essential
  - libffi-dev
  - libssl-dev
  - sshpass

Note: If the O/S is reinstalled or configured the system with valid repository that contains required packages, an Administrator would need to reinstall the packages.

Target Host Requirements

For the target host, supported systems include:

- Red Hat Enterprise Linux 7.5

In addition, the following is also required on the host:

- Configured access to user over ssh
- Full network connectivity between target hosts

Red Hat Enterprise Linux 7.5

Required packages include:

- byacc
- cifs-utils
- ebtables
- ethtool
- gcc
- gcc-c++
- git
- iproute
- iptables >= 1.4.21
- libcgroup
- libcgroup-devel
- libcgroup-tools
- libffi-devel
- libseccomp-devel
- libtool-ltdl-devel
- make
- nfs-utils
- openssh
- openssh-clients
- openssl
- openssl-devel
- policycoreutils-python
- python
- python-backports
- python-backports-ssl_match_hostname
- python-devel
- python-ipaddress
- python-setuptools
- rsync
- selinux-policy >= 3.13.1-23
- selinux-policy-base >= 3.13.1-102
- selinux-policy-targeted >= 3.13.1-102
- socat

- systemd-libs
- util-linux
- vim
- wget

Note: Deep Learning Studio Installer supports two modes: full installation or upgrade. Full installation requires that RedHat 7.5 O/S is installed and configured on the host account to the information included in this document. F

Configuration Tasks

The Intel DL Studio configuration tasks include configuring the files, proxies, and DNS servers and other configuration tasks.

Note: In the examples shown, **Green** indicates parameter name; and, **Blue** indicates exemplary parameter value.

Configuration File and Options

Proxy Configuration

- **Description:** This is the Proxy setting for internal applications.
- **Default value:** {}

Example

```
proxy:
http_proxy: http://<your proxy address and port>
https_proxy: http://<your proxy address and port>
no_proxy: .<localhost, any other addresses>, 10.0.0.1,localhost,.dlse
```

Note: Proxy address should be replaced by a specific value by a client.

dns_servers

- **Description:** This is a list of DNS servers used for resolution: a max of three (3) entries.
- **Default value:** []

Example

```
dns_servers:
  - 8.8.8.8
  - 8.8.4.4
```

dns_search_domains

Description: This is a domain used as part of a domain search list.

Default value: []

Example:

```
dns_search_domains:
  - example.com
  - example.org
```

domain

- **Description:** This is the *Internal* domain name.
- **Default value:** dlse

Example

- `domain: company`

nodes_domain

- **Description:** Internal subdomain for infrastructure. Full domain for an infrastructure is: `<nodes_domain>.<domain>`
- **Default value:** infra

Note: The *domain* setting is for top domain. Together with *nodes_domain* it creates a full zone domain. For example:

  - `domain: "dlse"`
  - `nodes_domain: "infra"`

This results in full domain `"infra.dlse"`.

Example

```
nodes_domain: lab007
```

Note: These IP addresses *should not* conflict with Internal IP address ranges.

k8s_domain

Description: This is the Internal subdomain for Kubernetes resources. Full domain for infrastructure is: `<k8s_domain>.<domain>`

Default value: kubernetes

Example

```
k8s_domain: kubernetes-001
```

kubernetes_pod_subnet

- **Description:** This is the Network Range for Kubernetes pods.
- **Default value:** 10.3.0.0/16

Example

```
kubernetes_pod_subnet: 10.128.0.0/16
```

kubernetes_svc_subnet

- **Description:** This is the Network Range for Kubernetes services.
- **Default value:** 10.4.0.0/16

Example

```
kubernetes_pod_subnet: 10.129.0.0/16
```

insecure_registries

- **Description:** This is a list of insecure docker registries added to configuration.
- **Default value:** []

Example

```
insecure_registries:
- my.company.registry:9876
```

enabled_plugins

- **Description:** This is a list of enabled Yum* plugins.
- **Default value:** []

Example

```
enabled_plugins:
- presto
```

disabled_plugins

- **Description:** This is a list of disabled Yum plugins.
- **Default value:** []

Example

```
disabled_plugins:
- presto
```

use_system_enabled_plugins

- **Description:** This defines if Yum should use system-enabled plugins.
- **Default value:** False

Example

```
use_system_enabled_plugins: True
```

enabled_repos

- **Description:** This is a list of enabled repositories, and is used for external dependencies installation.
- **Default value:** []

Example

```
enabled_repos:
- rhel
```

disabled_repos

- **Description:** This is a list of disabled repositories, and is used for external dependencies installation.
- **Default value:** []

Example

```
disabled_repos:
- rhel
```

use_system_enabled_repos

- **Description:** This defines if the default system repositories should be enabled in external dependencies installation.
- **Default value:** True

Example

```
use_system_enabled_repos: True
```

dls4e_configuration

- Description: Definition of the Intel DL Studio Applications configuration.
- Default value: {}

input_nfs

- **Description:** Definition of input NFS mounts for samba. By default, internal NFS provisioner is used.
- **Default value:** {}

Fields

- **path:** NFS path to mount
- **server:** NFS server

output_nfs

- **Description:** This is the definition of output NFS mounts for Samba. By default, internal NFS provisioner is used.
- **Default value:** {}

Fields

- **path:** NFS path to mount
- **server:** NFS server

Example

```
dls4e_configuration:
  input_nfs:
    path: /exports/test/input
    server: 10.0.0.1
  output_nfs:
    path: /exports/test/output
    server: 10.0.0.1
```

Features

- **Description:** This is a map of enabled features and more information can be found under: Feature Plugin Configuration, page 26.

Example

```
features:
  nfs: True
  redsocks: True
```

Examples

- Examples located at: Appendix A. Simple Configuration Example, page 33.

Inventory Tasks

This is a *yaml file* used by Ansible* for installation. It contains the system's inventory, which defines the hosts and groups of hosts upon which commands, modules, and tasks in a playbook operate. An admin should update it before starting the installation process. In addition, the content of this file should reflect hardware on which the Intel DL Studio software is going to be installed.

The file contains two sections:

- **Master:** contains a description of a master node. This section must contain *exactly* one row.
- **Worker:** contains descriptions of workers. Each worker is described in one row. In this section, it can have one or many rows depending on a structure of a cluster.

Format of rows with node's descriptions is as follows (all parts of such row are separated with spaces):
`[NAME] [VARIABLES]`
where:

- **NAME:** This is a name of a node. It should be a hostname of a real node (however, this *is not* a requirement)
- **VARIABLES:** Variables describing the node in the following format:
  - `[VARIABLE_NAME]=[VARIABLE_VALUE]`
  - List of available variables is given below. If certain variable has a default value, which satisfies a user's needs; however, it *does not* have to be added to a row. A row may contain more than one variable, in this case variables are separated with spaces.

Installation Inventory Variables

Basic ssh parameters for RedHat* Ansible* should be used to determine how to gain root access on remote hosts, including:

- ansible_ssh_user
- ansible_ssh_pass
- ansible_ssh_host
- ansible_ssh_port
- ansible_ssh_private_key_file

Note: If an Administrator decides to choose something other than root for Ansible ssh user, the user *must be* configured in sudoers file with NOPASSWD option.

internal_interface

Internal interface name: This is used for internal communication between Kubernetes processes and pods.

Specifications

- **Type:** string
- **Default:** None
- **Required:** True
- **Used when:** always
- **Exclusive with:** -
- **Acceptable values:** interface name

local_data_device

local data device: This is used by elastic search as storage.

Specifications

- **Type:** string
- **Default:** None
- **Required:** True
- **Used when:** for master nodes
- **Exclusive with:** -
- **Acceptable values:** path to block device

local_data_path

Local data path: This is used as the mountpoint for `local_data_device`.

Specification

- **Type:** string
- **Default:** /data
- **Required:** False
- **Used when:** storage_type is set to auto
- **Exclusive with:** -
- **Acceptable values:** block device name

data_interface

Data interface name: This is used for *ScaleIO data transfer and communication*. This can be used in the same way as: `internal_interface`.

Specification

- **Type:** string
- **Default:** None
- **Required:** True
- **Used when:** always

- **Exclusive with:** -
- **Acceptable values:** interface name

external_interface

external interface name: This is used for external network communication.

Specification

- **Type:** string
- **Default:** None
- **Required:** True
- **Used when:** always
- **Exclusive with:** -
- **Acceptable values:** interface name

Examples

Examples located in: Appendix B. Simple Inventory Example, page 34.

Installation Requirements

Intel® DL Studio Package

Extraction from Local Package

Untar the package from a local resource by doing the following:

- **Extract:** `tar -zxf dls4e-1.0.0-beta.tar.gz -C <destination>`

Structure

In the extracted archive, the following appears:

- **installer.sh:** sh definition of installation process
- **bin:** binary directory
- **dls4e:** DLS4 Enterprise applications deployer
- **platform:** kubernetes platform deployer
- **libs:** contains various scripts that are used by the installer
- **docs:** documentation

Components Version

To see the list of installed components and their versions, refer to: List of Required External Software Components, page 7.

Installation Process

Before proceeding with this step, you must create a configuration and an inventory file (see Appendix A. Simple Configuration Example, page 33 and Appendix B. Simple Inventory Example, page 34)

Installation Procedure

To install Intel DL Studio, follow these steps:

1. Export variables with configuration file, and:

- **ENV_INVENTORY (mandatory):** Inventory file location, for example:
  `export ENV_INVENTORY=$(realpath <inventory file>)`
- **ENV_CONFIG (mandatory):** Configuration file location, for example:
  `export ENV_CONFIG=$(realpath <config file>)`

2. Run the installation:
   `./installer.sh install`

Installer's Parameters

- **install:** kubernetes and DLS4Enterprise or Intel DL Studio
- **platform-install:** kubernetes installation
- **dls4e-install:** DLS4Enterprise installation
- **dls4e-upgrade:** DLS4Enterprise installation upgrade

Note 1: The default installation is targeting: `/opt/dls4e`. In addition, there are some binaries targeting: `/usr/bin` and settings located in `/etc/dls-cluster`. The path cannot be changed.

Note 2: The version is included in the package name.

Installation Build Process Artifacts

As an output of platform installation, files are created in the main installation directory:

`platform-admin.config - cluster admin config file`

As an output of the dls4e installation, files are created in main installation directory:

`dls4e-admin.config - DLS4E admin config file`

Upgrading Intel DL Studio

To upgrade the Intel DL Studio, do the following:

`export DLS_KUBECONFIG=/<PATH>/dls4e-admin.config`

- Call the installer with dls4e-upgrade:
  `./installer.sh dls4e-upgrade`

Note: It is recommended *to not use* the cluster during an upgrade.

Create a User Account

Creating a new user account creates a user account configuration file compliant in format with *kubectl* configuration files.

Setting Up Admin Account

Before creating admin accounts, you need to complete the following steps:

1. Install dlsctl based on your IT requirements.
2. Copy the dls4e-admin.config file to the machine where dlsctl resides. Place this configuration file at whatever <PATH> is convenient.
   - **Optional:** Copy the file, change the name of the file, and then modify the values within the file for the following fields.

     ```
     contexts:
     - context:
         user: new-user-name
     users:
       - name: new-user-name
     ```
3. Set up the KUBECONFIG variable to point to the full path of dls4e-admin.config, to where you copied the file in step 2. Follow the instructions below to create users.

   Execute: export `KUBECONFIG=<PATH>/dls4e-admin.config`

Creating a User

To create a user, perform these steps:

Note: The following is used to create a Deep Learning Studio user, not an Admin. In addition, a user *does not* have SSH access to the servers.

1. The dlsctl `user create` command sets up a namespace and associated roles for the named user on the cluster. It sets up *home* directories, named after the username, on the *input* and *output* network shares with file-system level access privileges.

   Execute: `dlsctl user create <username>`
2. This command also creates a configuration file named `<username>.config` and places this file in the user's home directory.

   Note 1: Users with the same name *cannot* be created directly after being removed (see Troubleshooting, page 28 for more details).

   Note 2: User names are limited to 32 characters maximum, must be lowercase, no underscores, periods, or special characters, and must start with a letter *not* a number. You can use a hyphen to join user names, for example: john-doe.
3. To verify that your new user has been created:

   Execute: `dlsctl user list`
4. This lists all users, including the new user just added, but *does not* show admins. An example is shown below.

```
| Name   | Creation date       | Date of last submitted job   |   Number of running jobs |   Number of queued jobs |
|--------+---------------------+------------------------------+--------------------------+-------------------------|
| jane   | 2018-09-17 08:32:16 | 2018-09-18 09:32:26          |                        1 |                       1 |
| john   | 2018-09-17 08:53:10 |                            _ |                        0 |                       0 |
```

5. As Admin, provide *username.config* file to the Intel DL Studio user. The user can save this file wherever it is convenient in their home directory:

   Execute: `cp <username>.config ~<PATH>/.`

6. Use the export command to set this variable for the user:

   Execute: `export KUBECONFIG=<PATH>/<USERNAME>.config`

Delete a User Account

Only an Administrator can delete user accounts and deleting a user removes that user's account from the Intel DL Studio software; therefore, that user *will not* be able to log in to the system. This will halt and remove all experiments and pods, however all artifacts related to that user's account, such as the users input and output folders and all data related to past experiments he/she submitted remains.

To remove a user:

Execute: `dlsctl user delete <user_name>`

This command asks for confirmation.

Do you want to continue? [y/N]: press y to confirm deletion.

To permanently remove (purge) all artifacts associated with the user and all data related to past experiments submitted by that user:

Execute: `dlsctl user delete <user_name> -p/--purge`

Note 1: No data from the Intel DL Studio storage is deleted during this operation.

Note 2: The `dlsctl user delete` command may take up to 30 seconds to delete the user. A new user with the same user name *cannot* be created until after the delete command confirms that the first user with the same name has been deleted (see Troubleshooting, page 28 for more details).

Launching the Web UI

To review the Resources Dashboard, launch the Web UI from the Command Line Interface (CLI).

Do the following to launch the Web UI:

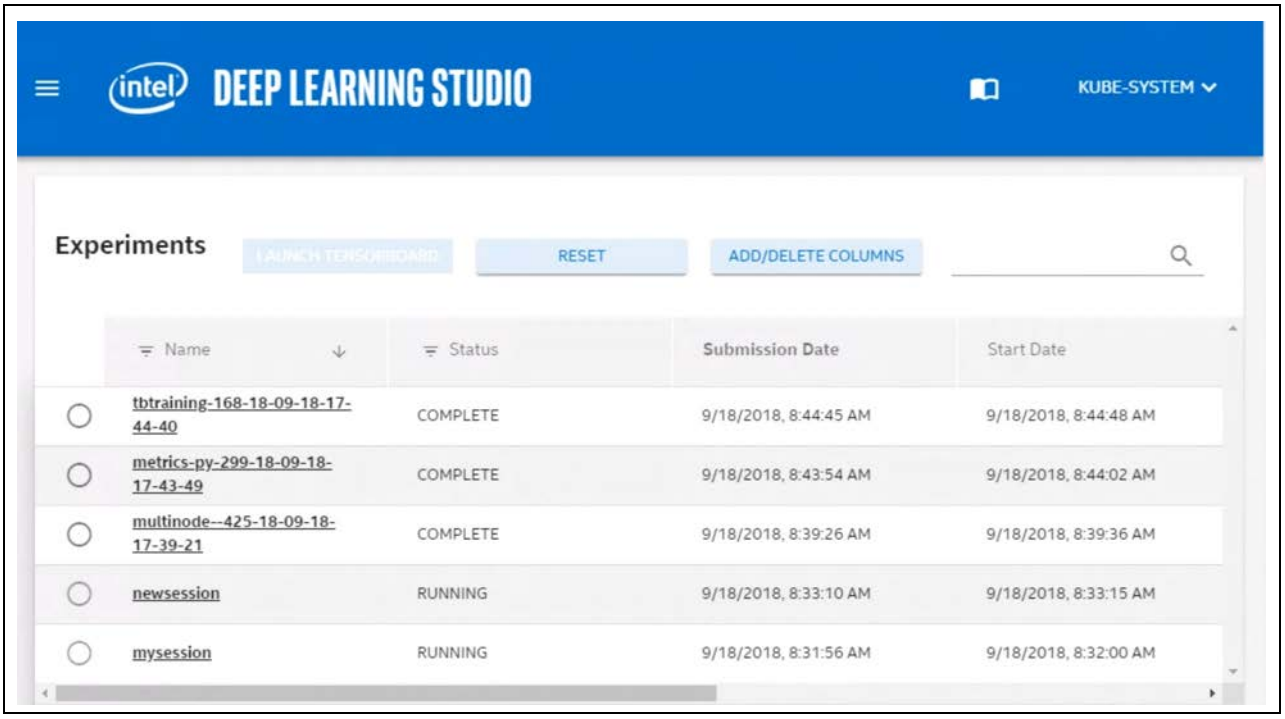
1. **Execute:** dlsctl launch webui --port 1080

   Note: The port shown above is an example only.

   The Web UI displays. For Admins, the Web UI displays empty an list experiments and shows the following message:

   No data for currently signed user. Click here to load all users data.

2. To view experiments *for all users*, click the displayed link.



Features and Network File System

Intel DL Studio features include NFS and Redsocks*.

Features List

- **NFS - default:** enabled
- **redsocks - default:** disabled

How to Enable Features

Additional features can be enabled using features object in configuration, as shown below.

Example

```
features:
  redsocks: False
```

Feature Plugin Configuration

Configuration for features should be placed under `features_config`.

Example

```
features:
  redsocks: True
features_config:
  redsocks:
    IP: my-socks5-proxy
    Port: 1080
```

Network File System Overview

The Network File System, or NFS allows a system to share directories and files with others over a network. The advantage of using NFS is that end-users as well as programs can access files on remote systems in the same way as local files. In addition, NFS uses less disk space, as it can be store data on a single machine while remaining accessible to others over a network.

Redsocks Configuration

Redsocks configuration is optional part of the installer; therefore, it might apply only to limited number of installations.

Configuration Parameters

IP

Description: This is the IP address of Socks5 proxy.

Required: True

Port

Description: This is the port address of Socks5 proxy.

Required: True

Interfaces

Description: Comma-separated list of interfaces from which traffic should be managed by RedSocks.

Required: False

Default: cni0

Troubleshooting

Jupyter Error

- Saving a file causes the following error:
  ```
  Creating file failed. An error occurred while creating a new file.
  Unexpected error while saving file: input/home/filname [Errno 2]
  No such file or directory: '/mnt/input/home/filename'
  ```

The error appears when a user tries to save file in `/input/home` folder which is a read only folder. In Jupyter, select the `/output/home` folder to correct this issue.

Note 1: Closing the Jupyter notebook window in a web browser causes experiments to stop executing. Attaching to the same Jupyter session still shows the stopped experiment.

Note 2: This is a known issue in Jupyter https://github.com/jupyter/notebook/issues/1647. Currently there *is no* workaround.

User Management Error

- Users with the same name *cannot* be created directly after its removal.

After deleting a user name and verifying that the user name *is not* on the list of user names, it *is not* possible to create a new user with the same name within short period of time. The reason is that the user's related Kubernetes objects are deleted asynchronously by Kubernetes and it can take some time.

Note: To resolve, wait a few minutes before creating a user with the same name.

Docker Error

- The Docker installation on the client machine takes up a lot of space and contains a lot of container images.

https://docs.docker.com states: *Docker takes a conservative approach to cleaning up unused objects (often referred to as "garbage collection"), such as images, containers, volumes, and networks: these objects are generally not removed unless you explicitly ask Docker to do so.*

Note: Refer to the following information for detailed instructions on how to prune unused Docker images: https://docs.docker.com/config/pruning/

DLSCTL Connection Error

- Launching tensorboard instance and launching WebUI do not work.

After running the dlsctl launch webui or the dlsctl launch tb <experiment_name> commands, a connection error message may be visible. During the usage of these commands, a proxy tunnel to the cluster is created. As a result, a connection error can be caused by an incorrect user-config generated by Administrator or by incorrect proxy settings in a local machine. To prevent this, be sure that a valid user-config is used and check proxy settings. In addition, ensure that the current proxy settings do not block any connection to a local machine.

Insufficient Resources Causes Experiments Failures

- Experiment fails just after submission even if script itself is correct.

If there is not enough resources on a Kubernetes cluster, the pods used in experiments are evicted. This results in failure of the whole experiment, even if there are no other reasons of this failure like those caused by users (like lack of access to data, errors in scripts and so on).

It is recommended that the Administrator investigate the failures to determine a course of action. For example, why all resources have been consumed and then try to free them.

Removal of Docker Images

- Due to known errors in Docker Garbage Collector making automatic removal of Docker images is laborious and error-prone.

Before running the Docker Garbage Collector, the administrator should remove images that are no longer needed, using the following procedure:

1. Expose the internal docker registry's API by exposing locally port 5000, exposed by *dls4enterprise-docker-registry service* located in the *dls4e* namespace. This can be done, for example by issuing the following command on a machine that has access to Intel DL Studio: `kubectl port-forward svc/dls4enterprise-docker-registry 5000 -n dls4e`
2. Get a list of images stored in the internal registry by issuing the following command (it is assumed that port 5000 is exposed locally): curl http://localhost:5000/v2/_catalog
3. From the list of images received in the previous step, choose those that should be removed. For each chosen image, execute the following steps:
   a. Get the list of tags belonging to the chosen image by issuing the following command: curl http://localhost:5000/v2/<image_name>/tags/list
   b. For each tag, get a digest related to this tag:
      ```
      curl --header "Accept:
      application/vnd.docker.distribution.manifest.v2+json"
      http://localhost:5000/v2/<image_name>/manifests/
      ```
      Digest is returned in a header of a response under the Docker-Content-Digest key
   c. Remove the image from the registry by issuing the following command:
      ```
      curl -X "DELETE" --header "Accept:
      application/vnd.docker.distribution.manifest.v2+json"
      http://localhost:5000/v2/<image_name>/manifests/
      ```
4. Run Docker Garbage Collector by issuing the following command: `kubectl exec -it $(kubectl get --no-headers=true pods -l app=docker-registry -n dls4e -o custom-columns=:metadata.name) -n dls4e registry garbage-collect /etc/docker/registry/config.yml`
5. Restart system's Docker registry. It can be done by deleting the pod with label: `dls4e_app_name=docker-registry`

Running Experiments are Evicted After Running Memory-consuming Horovod's Training

When running memory-consuming multi-node, Horovod-based experiments evicts other experiments that work on the same nodes where Horovod-related tasks were scheduled. As a result, such behavior is caused by Horovod-related tasks consume all memory on certain nodes which leads to eviction of other tasks running on these nodes. This is the standard behavior of Kubernetes.

To prevent such cases, schedule Horovod tasks to be only one on their nodes. It can be achieved by setting the requested number of CPUs when submitting an experiment near to maximum allocatable number of CPUs for cluster's nodes (if cluster has nodes with different numbers, it should be set to the greatest available number of CPUs on nodes. In this case, take into account also number of nodes involved in this experiment).

- Number of allocatable CPUs can be gathered by executing the kubectl describe node: <node_name> command.
- Number of allocatable CPUs is displayed in the `Allocatable->cpu` section.
- Number of CPUs gathered this way should be passed as: `-p resources.requests.cpu` <cpu_number> parameter while issuing experiment submit.

Command Connection Error

After running `dlsctl launch webui or dlsctl launch tb <experiment_name>` command connection error message may display. During this command, a proxy tunnel to the cluster is created. Connection errors can be caused by incorrect user config generated by an Administrator or incorrect proxy settings for local machine.

To prevent this, ensure that a valid user config is used and check proxy settings. In addition, ensure that current proxy settings do not block any connection to a local machine.

Inspecting Draft Logs

- While viewing the logs and output from some commands, a user may see the following message.

Inspect the logs with: `` `draft logs 01CVMD5D3B42E72CB5YQX1ANS5` ``

However, when running the command, the result is that the command *is not* found (or the logs *are not* found if there is a separate draft installation). This is because draft is located in DLSCTL config dir 'config' and needs a `--home` parameter.

To view the draft logs, use the following command:

```
$ ~/config/draft --home ~/config/.draft logs 01CVMD5D3B42E72CB5YQX1ANS5
```

Experiment's Pods stuck in Terminating state on Node Failure

There may be cases where a node suddenly becomes unavailable, for example due to a power failure. Experiments using TFJob templates, which were running on such a node, will stay Running in dlsctl and pods will be Terminating indefinitely. An experiment is not rescheduled automatically.

An example of such occurrence is visible using kubectl tool:

```
                    user@ubuntu:~$ kubectl get nodes
    NAME                              STATUS     ROLES    AGE   VERSION
    worker0.node.infra.dlse           NotReady   Worker   7d    v1.10.6
    worker1.node.infra.dlse           Ready      Worker   7d    v1.10.6
    master.node.infra.dlse            Ready      Master   7d    v1.10.6

                    user@ubuntu:~$ kubectl get pods
NAME                                  READY   STATUS        RESTARTS   AGE
experiment-master-0                   1/1     Terminating   0          45m
tensorboard-service-5f48964d54-tr7xf  2/2     Terminating   0          49m
tensorboard-service-5f48964d54-wsr6q  2/2     Running       0          43m
tiller-deploy-5c7f4fcb69-vz6gp        1/1     Running       0          49m
```

This is related to unresolved issues found in kubernetes and design of tf-operator:

- https://github.com/kubeflow/tf-operator/issues/720
- https://github.com/kubernetes/kubernetes/issues/55713

To solve this issue, manually resubmit the experiment using dlsctl.

A Multinode, Horovod-Based Experiment gets FAILED Status after Pod's Failure

If during execution of a multinode, Horovod-based experiment one of pods fails, then the whole experiment gets the FAILED status. Such behaviour is caused by a construction of Horovod framework. This framework uses an Open MPI framework to handle multiple tasks instead of using Kubernetes features. Because of this, it cannot rely also on other Kubernetes features like restarting pods in case of failures. This is why training jobs using Horovod don't resurrect failed pods. The Intel DL Studio system also doesn't restart such training jobs, so if a user encounters such a case - he/she should rerun the experiment manually. The construction of multinode tfjob-based experiments is different as it uses Kubernetes features. Thus, training jobs based on tfjobs restart failing pods so an experiment can be continued after their failure without a need to be restarted manually by a user.

Nodes Hang When Running Heavy Training Jobs

Running heavy training jobs on workers with the operating system kernel older than 4.* might lead to hanging the worker node. See https://bugzilla.redhat.com/show_bug.cgi?id=1507149

This may occur when a memory limit for 0 job is set to a value close to the maximum amount of memory installed on this node. These problems are caused by errors in handling memory limits in older versions of the kernel. To avoid this problem, it is recommended to install on all nodes of a cluster with a newer version of a system's kernel.

The following kernel was verified as a viable fix to the issue:

https://elrepo.org/linux/kernel/el7/x86_64/RPMS/kernel-ml-4.19.2-1.el7.elrepo.x86_64.rpm

To install the new kernel please reference chapter 5 "Manually Upgrading the Kernel" in RedHat's Kernel Administration Guide documentation https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/kernel_administration_guide/ch-manually_upgrading_the_kernel

Note: The above kernel does not include RedHat's optimizations and hardware drivers.

Platform Client for Windows

Using "standard" windows terminals (cmd.exe, power shell) is enough to interact with platform, but there is a sporadic issue with the output. Some lines can be surrounded with incomprehensible characters, for example:

```
[K[?25hCannot connect to K8S cluster: Unable to connect to the server: d...
```

Please note that the recommended shell environment for Windows operating system is bash. For bash-based terminals, this issue does not occur.

Appendix A. Simple Configuration Example

```
---
proxy:
  http_proxy: http://< your proxy address and port>:911
  https_proxy: http://<your proxy address and port>:912
  no_proxy: .< localhost, any other addresses>,127.0.0.1,localhost,.dlse

dns_servers:
  - 10.102.60.20
  - 10.102.60.30

dls4e_configuration:
  input_nfs:
    path: /exports/test/input
    server: 10.91.120.230
  output_nfs:
    path: /exports/test/output
    server: 10.91.120.230
```

Note: Proxy addresses and ports should be replaced by a specific value for a client.

Appendix B. Simple Inventory Example

```
[master] vt-000-01 ansible_ssh_host=10.91.120.60 ansible_ssh_user=cloud-user
internal_interface=ens3 data_interface=ens3 external_interface=ens4


[worker] vt-000-02 ansible_ssh_host=10.91.120.61 ansible_ssh_user=cloud-user
internal_interface=ens3 data_interface=ens3 external_interface=ens4 vt-000-04
ansible_ssh_host=10.91.120.63 ansible_ssh_user=cloud-user
internal_interface=ens3 data_interface=ens3 external_interface=ens4 vt-000-05
ansible_ssh_host=10.91.120.64 ansible_ssh_user=cloud-user
internal_interface=ens3 data_interface=ens3 external_interface=ens4 vt-000-06
ansible_ssh_host=10.91.120.108 ansible_ssh_user=cloud-user
internal_interface=ens3 data_interface=ens3 external_interface=ens4 vt-001-10
ansible_ssh_host=10.91.115.59 ansible_ssh_user=cloud-user
internal_interface=ens3 data_interface=ens3 external_interface=ens4 vt-001-11
ansible_ssh_host=10.91.115.61 ansible_ssh_user=cloud-user
internal_interface=ens3 data_interface=ens3 external_interface=ens4 vt-001-12
ansible_ssh_host=10.91.115.65 ansible_ssh_user=cloud-user
internal_interface=ens3 data_interface=ens3 external_interface=ens4
```