OMNI Platform 1.11.0

Administrator Guide

Contents

1. System Requirements

1.1. Minimum System Requirements

To deploy OMNI Platform, make sure that your machine meets the following requirements:

- CPU with 4+ cores 3+ GHz and AVX/AVX2 support, for example, Intel® Xeon® E3-1220v5
- 16GB RAM
- SSD with 100+ GB free space and 20%+ available space in the file system

If you need to deploy a separate database server (by default the database server is launched at the machine where the Platform is deployed), the machine for database server deployment should meet the requirements below:

- CPU (2 cores)
- 8GB RAM
- PostgreSQL >= 11.2
- SSD/HDD 100GB

The software below should be installed on your machine:

- OC Ubuntu 20.04.4
- Docker 20.10.17
- Kubernetes 1.23.8
- Helm 3.9.1

To install Docker, Kubernetes and Helm, use on_premise/setup/install-packages.sh script supplied with the distribution kit (the Internet connection is required).

Note: The minimum system requirements listed in this section are mostly used for testing the basic functionality of OMNI Platform. For more accurate system configuration based on your needs, please contact 3DiVi technical experts.

1.2. GPU Requirements (optionally)

Nvidia Container Toolkit v1.11.0

Graphics card: Nvidia GTX 1080 Ti or higher with CUDA 10.2 support.

1.3. Docker Settings for GPU Usage (optionally)

1.3.1 Docker Configuration

To set `nvidia-container-runtime` as the default low-level runtime, add the following lines to the configuration file located at `/etc/docker/daemon.json`:

```
"default-runtime": "nvidia",
"runtimes": {
    "nvidia": {
        "path": "/usr/bin/nvidia-container-runtime",
        "runtimeArgs": []
    }
}
```

1.3.2 Applying Configuration

To apply configuration, restart docker-service using the command below:

```
$ sudo systemctl restart docker
```

2. Deployment

2.1. OMNI Platform Update

If you already use OMNI Platform on-premise and get the new version, follow the instructions and commands below to update OMNI Platform, otherwise skip this section.

1. Move to `on_premise` folder and stop OMNI Platform:

```
$ ./setup/uninstall-platform.sh
```

2. Make sure that all service containers are stopped:

```
$ watch 'kubectl get pods'
```

Service statuses should change from *Running* to *Terminating*. As a result, all services are removed from the console output.

3. Delete the deployed Kubernetes cluster:

```
$ sudo kubeadm reset
```

4. Delete auxiliary files of Kubernetes cluster:

```
$ sudo rm -rf ~/.kube/
```

5. Reset IPVS tables of your system:

```
$ sudo ipvsadm --clear
```

Note 1: When the Platform is deleted, the entire database is saved in the `/kv/pgdata` directory. For further use of this database you need to specify the same authorization data and database name when installing the new version of the Platform. Otherwise, delete the `/kv/pgdata` folder `(sudo rm -rf /kv/pgdata)` to create a new database during Platform deployment.

Note 2: After updating OMNI Platform, you'll need to reset the platform settings, unless you save them before updating. In the latter case, after updating, that will be enough to replace the files with default settings with files with saved values (if the fields in the old and new files completely match). The settings represent the values of the environment variables specified in the `./setup/settings.env` file (See section 2.2.3 for details), as well as the scalability settings specified in the `./deploy/values.yaml` file (See section 2.4.4 for details).

2.2. Upload Images and Create a Kubernetes Cluster

2.2.1 Preparation

Download and extract OMNI Platform distribution kit to the machine used for installation. You'll get the link to the distribution kit to your email.

Open a system console, move to `on-premise` directory of the distribution kit and check folder contents by running a command:

```
$ find -maxdepth 1
```

As a result, files and folders from the distribution kit will be shown in the console:

```
./deploy
./ingress-nginx-4.2.0.tgz
./OMNIAgent_Linux_x64.run
./OMNIAgent_Windows_x64.exe
./integration_tests
./kube-flannel.yml
./kubeadm.yaml
./license_server
./nvidia-device-plugin-0.12.2.tgz
./pdf_docs
./platform_images.tar.gz
./setup
./upload_script
```

Contents of the distribution kit:

- `./pdf_docs/administrator_guide.pdf` is an administrator guide to deploy, test and debug OMNI Platform;
- `./pdf_docs/user_guide.pdf` is a user guide to start up and use OMNI Platform;
- `./pdf_docs/integration_api.pdf` is OMNI Platform API reference;
- `./pdf_docs/release_notes.pdf` is release notes with information about new features, bug fixes and improvements;
- `./pdf_docs/agent_user_guide.pdf` is a separate user guide for OMNI Agent;
- `./OMNIAgent_Linux_x64.run` and `./OMNIAgent_Windows_x64.exe` are OMNI Agent installation files for Windows and Linux;
- `./license_server` are files required for launching a license server for OMNI Platform licensing;
- `./integration_tests` are scripts for OMNI Platform automatic testing after deployment;

- `./setup/settings.env` is a configuration file of OMNI Platform instance;
- `./upload_script` is a folder that contains an image upload script for creating profiles on the Platform from the dataset images.

Further commands are to be executed in the system console from the on_premise directory.

2.2.2 Upload Images

Upload images from the archive to the local registry:

```
$ sudo docker load -i platform_images.tar.gz
```

Uploading can last for about 5 minutes.

2.2.3 Enter Environment Variables

Open a configuration file `./setup/settings.env` in a text editor and set values for the following variables:

- MASTER_NODE_IP_ADDRESS: IP address of the machine which OMNI Platform is deployed to. You can get an IP address from your system administrator;
- DOMAIN root domain name: After deployment the access to OMNI Platform web interface and API is provided via `http://<DOMAIN>`. IP address for domain name `<DOMAIN>` should be configured at DNS server (See more details at 2.4.2 of this guide);
- RABBIT_USER and RABBIT_PASSWORD: Username and password to get access to a message broker, used for internal interaction of OMNI Platform services. Set an arbitrary name that consists of Latin letters without spaces and a password that consists of Latin letters and numbers without spaces;
- POSTGRES_USER, POSTGRES_PASSWORD and POSTGRES_DB: Database connection settings. At first deployment, set an arbitrary user name and database name that consists of Latin letters without spaces, and generate a password that consists of Latin letters and numbers. The database will be created automatically;
- SERVICE_KEY: A private key used for internal interaction of OMNI Platform services. Generate an arbitrary line that consists of Latin letters and numbers without spaces;
- LIC_KEY: A license key. This key is usually sent in the accompanying letter to the distribution kit. To get the key, contact your sales manager. **Note**: a license key is not required for trial period;
- PLATFORM_ADMIN_EMAIL and PLATFORM_ADMIN_PASSWORD: Credentials used to get access to OMNI Platform administrator web interface. At first deployment the system automatically creates a user with administrator privileges. Enter a valid email and

generate a password that consists of Latin letters and numbers, at least 8 characters long;

- PLATFORM_DEFAULT_EMAIL, PLATFORM_DEFAULT_PASSWORD: User credentials for accessing OMNI Platform web interface. The user will be created automatically at first deployment. Enter a valid email and generate a password consisting of Latin letters and numbers, at least 8 characters long;
- EMAIL_HOST, EMAIL_PORT, EMAIL_HOST_USER, EMAIL_HOST_PASSWORD, EMAIL_USE_SSL: SMTP server access settings. SMTP server is used to send emails for password reset, notifications, etc. To disable email sending, leave these fields blank. To obtain SMTP server access parameters, contact your network administrator. To enable/disable SSL protocol, set the EMAIL_USE_SSL parameter to true/false, respectively. Initially, this parameter has no value specified, which means “false” value by default;
- EMAIL_FROM : The value sent in FROM header and displayed as an email sender. The format requirements for this field may vary depending on SMTP server. An example of a FROM field value - `"Bob Example" <bob@example.org>`;
- QUERY_LIMIT : The value limits the number of returned elements in API requests to get system objects. Increasing this limit is not recommended, as API request run time may increase in several times. Also, please note that increasing the limits will lead to the system degradation.
- ENABLE_PROFILE_AUTOGENERATION: Auto-generation of profiles for incoming activities from the agent. Please note that enabling this option will increase the consumption of license resources (database size). If this function is not required, leave the field empty, otherwise set the value to 1.
- USE_CUDA: Responsible for use of CUDA cores in services for image processing. 0 - GPU disabled, 1 - GPU enabled for processing service.

Ensure that you saved the file after editing.

2.2.4 Install and Configure a Cluster

Note: If you already have a deployed cluster, go to section 2.2.6.

To create and configure the cluster, run the following command:

```
$ ./setup/init-cluster.sh
```

This command initializes a node for cluster deployment, creates secrets and necessary folders and installs ingress controller and nvidia-device-plugin, if graphics card is enabled.

2.2.5 Cluster Health Check

After initializing the master node, make sure that all nodes are ready for operation and have the *Ready* status. You can check this by running the command below:

```
$ kubectl get nodes
```

As a result, the following output will be displayed in the terminal:

```
NAME          STATUS     ROLES                   AGE     VERSION
master-node   Ready      control-plane,master    11d     v1.23.8
```

To check all cluster components, run the following command:

```
$ kubectl get all --all-namespaces
```

2.2.6 Using Local Database

Note: Local database is used by default. To connect the external database, go to section 2.2.7. To start using the local database, make sure that the server, where the database will be deployed, contains `/kv/pgdata` directory, otherwise create this directory using the commands below:

```
$ sudo mkdir -p /kv/pgdata
$ sudo chmod -R 777 /kv/pgdata
```

2.3. Configure Licensing

Note: If you have updated OMNI Platform using the commands from section 2.1, proceed directly to section 2.4.

2.3.1 Trial activation

Please note that:

- the Internet connection is required;
- running OMNI Platform on a virtual machine is not allowed.

The trial period is activated the first time you launch OMNI Platform. To activate the trial period, go to section 2.4.

2.3.2 Install a License Server

Before installation, open the `./setup/settings.env` file and set the IP address of the machine, on which the license server will be installed, in the LICENSE_SERVER_IP_ADDRESS variable.

If the license server is running on the same machine where the cluster is deployed, LICENSE_SERVER_IP_ADDRESS will be the same as MASTER_NODE_IP_ADDRESS.

Run the command to install and start the license server:

```
$ ./setup/install-lic-server.sh
```

Check that the license server is in the *running* status:

```
$ ./setup/status-license-server.sh
```

Console output example:

```
floatingserver.service - Floating License Server
     Loaded: loaded (/etc/systemd/system/floatingserver.service;
enabled; vendor preset: enabled)
     Active: active (running) since Tue 2022-12-20 12:25:54 +05; 1min
48s ago
```

To check that the license server is available, follow http://<LICENSE_SERVER_IP_ADDRESS>:8090 in your web browser. As a result, you should be redirected to the login form.

2.3.3 Online License Activation

If the machine, on which the license server is installed, does not have Internet connection, skip this section and go to section 2.3.3.

Before activation, make sure that the LIC_KEY variable (from file `./setup/settings.env`) contains the license key.

Run the license activation command:

```
$ ./setup/activate-lic-server.sh
```

When license is successfully activated, the console will return the following result:

```
[2022-12-20 12:25:53+05:00] INF Activating license key...
[2022-12-20 12:25:54+05:00] INF License activated successfully!
```

After activation is successfully completed, go to section 2.4.

2.3.4 Offline License Activation

Before activation, make sure that the LIC_KEY variable (from file `./setup/settings.env`) contains the license key.

Run the command below to generate an offline license request:

```
$ ./setup/activate-lic-server.sh --generate-offline
```

As a result, the `request-offline.license` file should appear in the `on_premise` directory.

Send the generated `request-offline.license` request file to ***support-platform@3divi.com***. The license file will be submitted in the response email.

Copy the received license file to the `on_premise` folder.

Open the configuration file `./setup/settings.env` in a text editor and fill in the variable `OFFLINE_LICENSE_FILE` with the license file name and its extension, if present, separated by a dot.

Run the license activation command:

```
$ ./setup/activate-lic-server.sh --activate-offline
```

When license is successfully activated, the console will return the following result:

```
[2022-09-08 01:30:36+05:00] INF Offline activating license key...
[2022-09-08 01:30:36+05:00] INF License activated successfully!
```

2.4. OMNI Platform Deployment

2.4.1 Launch Deployment

To deploy OMNI Platform in the cluster, run the following script:

```
$ ./setup/deploy.sh
```

To monitor the deployment progress, open another terminal tab and enter the following command:

```
$ watch 'kubectl get pods'
```

OMNI Platform is running if all pods have the *Running* status.

2.4.2 Configure DNS Server

To provide access to OMNI Platform, DNS server on your network should contain a record that `<DOMAIN>` domain is available at `<MASTER_NODE_IP_ADDRESS>`. Variable values can be obtained from `./setup/settings.env` file filled in in section 2.2.3. To complete this configuration, contact your network administrator.

For testing you need to fill in the IP address and domain in the `/etc/hosts` file on Linux or `C:\Windows\System32\drivers\etc\hosts` on Windows.

To do this, add a new line like `<MASTER_NODE_IP_ADDRESS> <DOMAIN>` at the end of this file, set the values for the corresponding variables and save the file. Note that you need to have administrator privileges to edit the `hosts` file.

To use OMNI Platform at the machine where it was deployed, you can use a script below. It will automatically add the necessary entry to the `/etc/hosts file`.

```
$ ./setup/add-dns.sh
```

2.4.3 Description of Deployed System

To get the status of OMNI Platform services, run the following command:

```
$ kubectl get pods
```

As a result, the console will display a list of services, their statuses, the number of restarts, and the service age. The example of console output:

```
NAME                                 READY   STATUS     RESTARTS     AGE
backend-dep-5478dd8d88-lfjp8         1/1    Running   0             18h
broker-dep-697f74bbbd-qqx5v          1/1    Running   0             18h
cache-dep-589c74b79f-kfblz           1/1    Running   0             18h
db-dep-8669dd8c47-mmltv              1/1    Running   0             18h
gateway-dep-67dcf75646-6vwlw         1/1    Running   0             18h
image-api-dep-79df7ff5cc-fmmdp       1/1    Running   0             18h
matcher-dep-7cd8fc9849-j2qnw         1/1    Running   0             18h
processing-dep-775bdf5875-kxrnq      1/1    Running   0             18h
quality-dep-7c47d85787-2qn58         1/1    Running   2 (18h ago)  18h
redis-dep-5d8cd4d657-88rzf           1/1    Running   0             18h
licensing-dep-567f4c64f-9wdzr        1/1    Running   0             18h
```

Overview of the services is given below:

- `backend-dep` is the main container of OMNI Platform, responsible for how most of API works;
- `db-dep` is an instance of PostgreSQL database that stores all information of OMNI Platform;
- `gateway-dep` is the nginx service, responsible for access to OMNI Platform and for the operation of OMNI Platform web interface;
- `matcher-dep` is the service responsible for searching a person in the database;
- `processing-dep` is the service responsible for person detection in the image with subsequent creation of a search template. This template is used when executing a request for face detection;
- `quality-dep` is the service responsible for calculating the image quality;
- `image-api-dep` is the service responsible for operation of ImageAPI available at the /image-api/ URL;
- `licensing-dep` restricts platform operation according to the license parameters.

2.4.4 Scalability

When the load increases, the following services can be scaled in manual mode to stabilize OMNI Platform operation:

- processing-dep
- quality-dep
- backend-dep
- gateway-dep
- ingress-nginx-controller

These services are described in section 2.4.3

To scale the service, run the command below:

```
$ kubectl scale deployment <SERVICE_NAME> --replicas <COUNT>
```

where `SERVICE_NAME` is a service name (for example, gateway-dep), and `COUNT` is a number of service instances.

Note: To scale ingress-nginx-controller service, add the “-n ingress-nginx” argument to the end of the command.

For successful scaling follow the information below:

1. To process a greater number of concurrent requests, you should scale `backend-dep`, `ingress-nginx-controller` and `gateway-dep`. The number of service instances is specified according to the formula: `<REQUESTS>/<CPU_COUNT>`, where `<REQUESTS>` is the desired number of concurrent requests, and `<CPU_COUNT>` is the number of logical CPU cores.
2. If most of the requests are related to image processing, scale `processing-dep` and `quality-dep` services. The number of service instances should not exceed the number of physical CPU cores.

For example:

To keep the load of A requests/sec for image processing, on a server with a physical number of CPU cores equal to B and a logical number of CPU cores equal to C, the services should be scaled as follows:

- processing-dep - min(A, B) instances
- quality-dep - min(A, B) instances
- backend-dep - A/C instances
- gateway-dep - A/C instances
- ingress-nginx-controller - A/C instances

To save the scaling settings, open the `./deploy/values.yaml` file, find the `service_replicas` module, and set the services to the newly selected values.

For the next installations of the platform, services will be automatically scaled to the specified values.

2.4.5 Database Backup and Restore

Note: Not supported for external databases.

To back up the database, run the command below:

```
$ ./setup/db-backup.sh <dump_path>
```

To restore the database, run the following command:

```
$ ./setup/db-restore.sh <dump_path>
```

`dump_path` - path to the database dump.

3. Functional Testing and Debugging

3.1 Functional Testing

The distribution kit contains a script for automatic check of OMNI Platform functionality. Specify the Platform URL `http://<DOMAIN>`, user email `<PLATFORM_DEFAULT_EMAIL>` and run the command:

```
$ python3 integration_tests/main.py <platform url> <user email> --show-trace
```

After running the script, the console will ask for a password. Enter the user password `<PLATFORM_DEFAULT_PASSWORD>` and press the *Enter* key. After the tests are completed successfully, the console will display the following text:

```
Test data prepared successfully
----------------------------------------------------------------------
-------------------------------
create_search_profile test started
create_search_profile test succeeded
----------------------------------------------------------------------
-------------------------------
Test data deleted successfully
```

3.1.1 Possible Testing Errors and Solutions

When testing errors occur, the system returns the following result:

```
Error: <error type>
Error message: <error message>
```

`Error` indicates the type of error that occurred, and `error message` provides more specific information about the error.

Commands for debugging services are described in section 3.2, the list of services and their responsibilities are given in section 2.4.3.

Combinations of errors and messages with possible solutions are listed below.

ConnectionError:

- **<urlopen error Wrong url format: asdasd>**
  Invalid URL format, please enter a valid address.

- **<urlopen error [Errno -2] Name or service not known> \ <urlopen error [Errno 111] Connection refused>**
  URL belongs to an unavailable service. Check the entered address, and make sure that OMNI Platform is deployed correctly and is accessible from the outside. If you have access by domain, check that /etc/hosts file has the domain that points to the IP address of the deployed OMNI Platform.

- **HTTP Error 405: Not Allowed**
  Make sure that the URL you entered will take you to OMNI Platform and not to a third-party service.

- **HTTP Error 502: Bad Gateway / HTTP Error 503: Service Temporarily Unavailable**
  Make sure the backend-dep service is deployed.

PlatformError:

- **connection to server at "localhost" (::1), port 5432 failed: Connection refused Is the server running on that host and accepting TCP/IP connections?**
  Make sure the database is available and working properly.

- **Authorization error**
  Make sure you enter the correct user password and email.

- **Wrong answer from server. JSON can not decoded**
  Make sure that the URL you entered takes you to OMNI Platform and not to a third-party service.

- **License has not been leased yet**
  Make sure the license server is running and OMNI Platform has access to it. Also, check that the license is activated correctly.

- **Low quality photo**
  Check that the service responsible for calculating the quality of photos is available and working correctly.

- **Profile not searched**
  Make sure that the service responsible for searching the database of profiles is available and works correctly.

If you have difficulty with the above errors or meet any other errors or messages that cannot be debugged and resolved on the spot, please contact our Tech Support Team.

3.2 Debugging

If the service doesn't work correctly, you can get the logs by running the command below, where you need to set the name of the pod you're interested in:

```
$ kubectl logs $POD_NAME
```

You can also get the logs of the previous launch attempt:

```
$ kubectl logs $POD_NAME --previous
```

Or through the Events section in the output of the command below:

```
$ kubectl describe pod $POD_NAME
```

4. Get System Access Data

To work with OMNI Platform, the user needs to have credentials to sign in the web interface, the Platform URL, and the API access token. How to obtain this information is described below.

4.1. Get User Password and Email

The user password and mail can be found in `settings.env` Platform configuration file in the `PLATFORM_DEFAULT_PASSWORD` and `PLATFORM_DEFAULT_EMAIL` variables, respectively.

4.2. Get URL of Deployed Server

OMNI Platform deployment domain is specified in `settings.env` file in `DOMAIN` variable.

As a result, access to the Platform can be obtained by URL: `http://<DOMAIN>`

For example: `http://your-company.com`

4.3. Get Access Token

To get API access token for a user registered with `PLATFORM_DEFAULT_EMAIL`, run the command below:

```
$ ./setup/get-token.sh
```

You can also open OMNI Platform web interface in web browser using URL from the previous section. Sign in to the Platform using the user credentials from section 4.1, follow the Platform API link in *Resources* panel of the home page, and submit the following request in GraphQL console:

```
query{
  me {
    workspaces {
      accesses {
        id
      }
    }
  }
}
```

As a result, GraphQL returns the response with access token:

```
{
  "data": {
    "me": {
      "workspaces": [
        {
          "accesses ": [
            {
              "id": "3460a1d7-214c-48c3-b2be-aa6d2a6bca09"
            }
          ]
        }
      ]
    }
  }
}
```

API access token is given in the id field.

5. Helpful Commands

Starting up the platform:

```
$ ./setup/deploy.sh
```

Stopping the platform:

```
$ ./setup/uninstall-platform.sh
```

Starting the license server:

```
$ sudo service floatingserver start
```

Stopping the license server:

```
$ sudo service floatingserver stop
```

6. Troubleshooting

Error occurred when generating offline license request

Problem: when executing a command `./setup/activate-lic-server.sh --generate-offline` you can get the following error:

```
ERR Missing file path for offline activation request file! Specify path using
'--offline-request' option.
```

Solution: ensure that the file `./setup/settings.env` contains a license key in `LIC_KEY` variable and a license server address in `LIC_SERVER_URL` variable.

Error occurred when installing Docker，Kubernetes and Helm

Problem: when executing a script command `on_premise/setup/install-packages.sh` you can receive the following error:

```
E: Sub-process /usr/bin/dpkg returned an error code (1)
```

Solution 1: the error can be caused by a corrupted dpkg database. In this case, reconfigure the dpkg package manager with the command:

```
$ sudo dpkg --configure -a
```

Solution 2: if errors appear during the installation of software packages, you can force the installation of the package using `-f` option:

```
$ sudo apt install -f
OR
$ sudo apt install --fix-broken
```

Options `-f` and `--fix-broken` can be interchangeably used to fix broken dependencies resulting from an interrupted package download.

Solution 3: if the first two solutions did not fix the problem, you can remove or purge the problematic software package as shown below:

```
$ sudo apt remove --purge package_name
```

Solution 4: You can also manually remove all files associated with the problematic package by running the command below. The files are located in the directory **`/var/lib/dpkg/info`**.

```
$ sudo ls -l /var/lib/dpkg/info | grep -i package_name
```

After listing the files, you can move them to the **/tmp** directory as shown below:

```
$ sudo mv /var/lib/dpkg/info/package-name.* /tmp
```

Alternatively, you can use the `rm command` to manually remove the files.

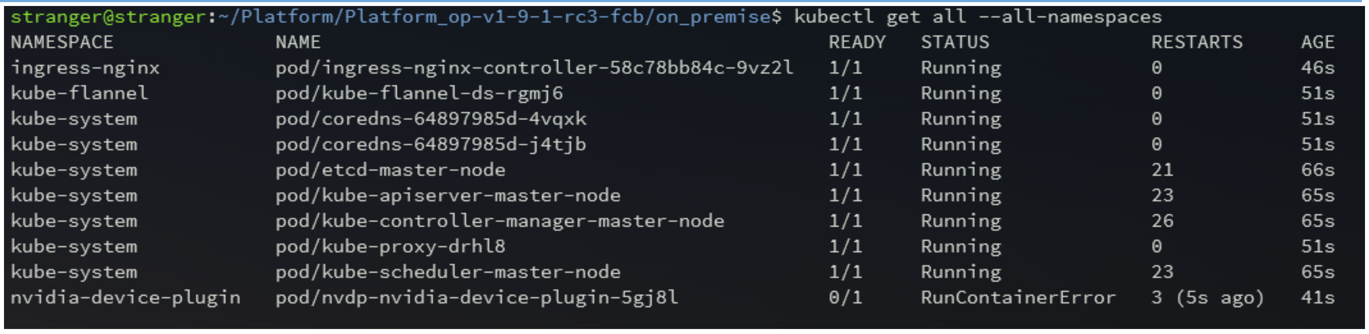
```
$ sudo rm -r /var/lib/dpkg/info/package-name.*
```

Error with `nvidia-device-plugin` when checking cluster components

Problem: when executing a command `kubectl get all --all-namespaces` you can receive the following error:

```
Error: failed to start container "nvidia-device-plugin-ctr": Error response from daemon: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: error during container init: error running hook #0: error running hook: exit status 1, stdout: , stderr: Auto-detected mode as 'legacy'
```

```
nvidia-container-cli: initialization error: nvml error: driver/library version mismatch: unknown
```



```
stranger@stranger:~/Platform/Platform_op-v1-9-1-rc3-fcb/on_premise$ kubectl get all --all-namespaces
NAMESPACE            NAME                                          READY   STATUS              RESTARTS     AGE
ingress-nginx        pod/ingress-nginx-controller-58c78bb84c-9vz2l   1/1     Running             0            46s
kube-flannel         pod/kube-flannel-ds-rgmj6                     1/1     Running             0            51s
kube-system          pod/coredns-64897985d-4vqxk                   1/1     Running             0            51s
kube-system          pod/coredns-64897985d-j4tjb                   1/1     Running             0            51s
kube-system          pod/etcd-master-node                          1/1     Running             21           66s
kube-system          pod/kube-apiserver-master-node                1/1     Running             23           65s
kube-system          pod/kube-controller-manager-master-node       1/1     Running             26           65s
kube-system          pod/kube-proxy-drhl8                          1/1     Running             0            51s
kube-system          pod/kube-scheduler-master-node                1/1     Running             23           65s
nvidia-device-plugin pod/nvdp-nvidia-device-plugin-5gj8l           0/1     RunContainerError   3 (5s ago)   41s
```

Solution:

1. For information about your graphics card and available drivers, run the following command:

```
ubuntu-drivers devices
```

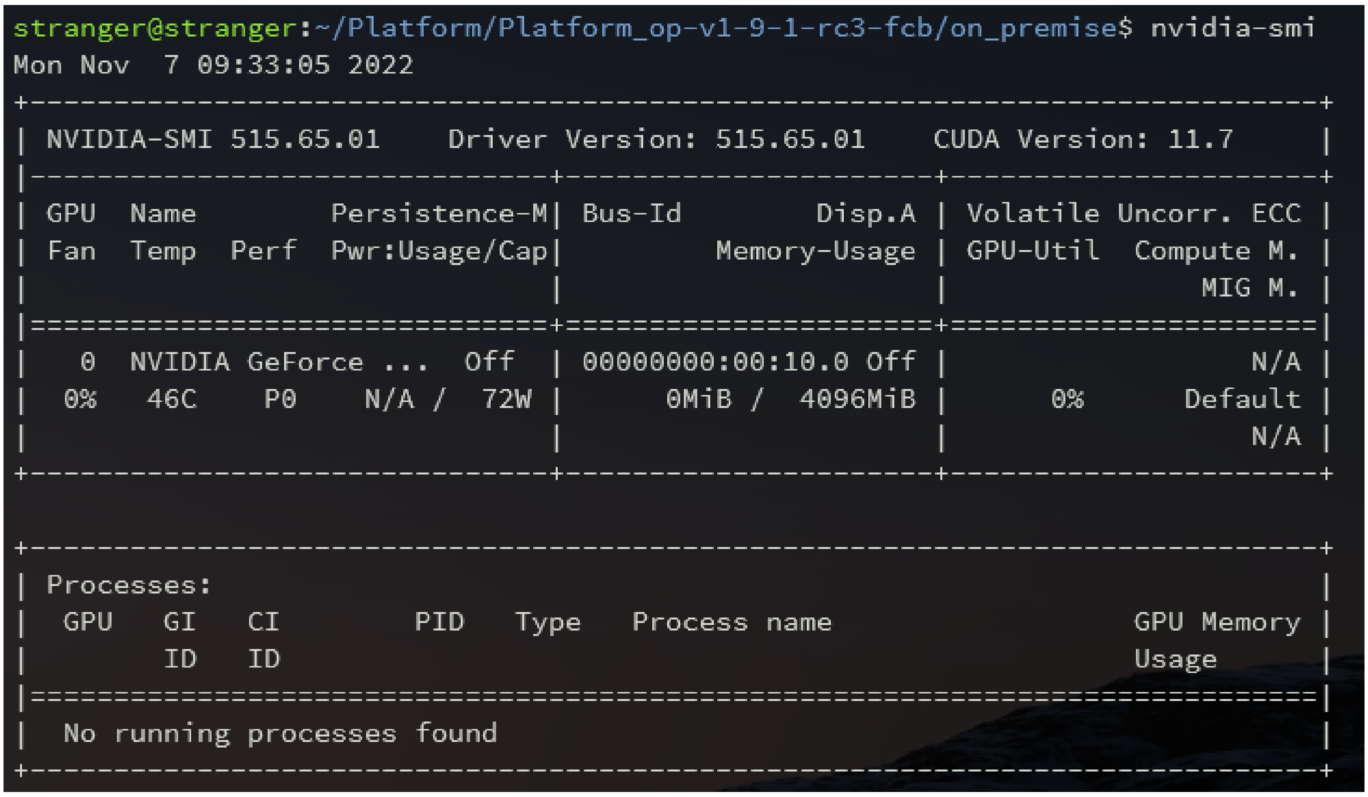
2. The console output indicates that the system has a "GeForce GTX 1050 Ti" graphics card and the recommended driver "nvidia-driver-515".

```
 == /sys/devices/pci0000:00/0000:00:10.0 ==
modalias : pci:v000010DEd00001C82sv00001458sd00003764bc03sc00i00
vendor   : NVIDIA Corporation
model    : GP107 [GeForce GTX 1050 Ti]
manual_install: True
driver   : nvidia-driver-510-server - distro non-free
driver   : nvidia-driver-450-server - distro non-free
driver   : nvidia-driver-390 - distro non-free
driver   : nvidia-driver-520 - distro non-free
driver   : nvidia-driver-418-server - distro non-free
driver   : nvidia-driver-515-server - distro non-free
driver   : nvidia-driver-515 - distro non-free recommended
driver   : nvidia-driver-510 - distro non-free
driver   : nvidia-driver-470-server - distro non-free
driver   : nvidia-driver-470 - distro non-free
driver   : xserver-xorg-video-nouveau - distro free builtin
```

3. To install the recommended driver, run the command:

```
sudo apt install nvidia-driver-515
```

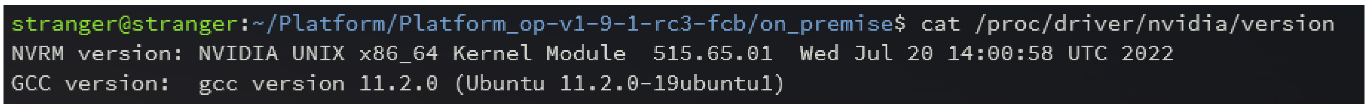
4. After installing the driver, you can view the status of the graphics card using the `nvidia-smi` monitoring tool:



```
stranger@stranger:~/Platform/Platform_op-v1-9-1-rc3-fcb/on_premise$ nvidia-smi
Mon Nov  7 09:33:05 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 515.65.01    Driver Version: 515.65.01    CUDA Version: 11.7     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  Off  | 00000000:00:10.0 Off |                  N/A |
|  0%   46C    P0    N/A /  72W |      0MiB /  4096MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

5. You can view the driver version using the command:

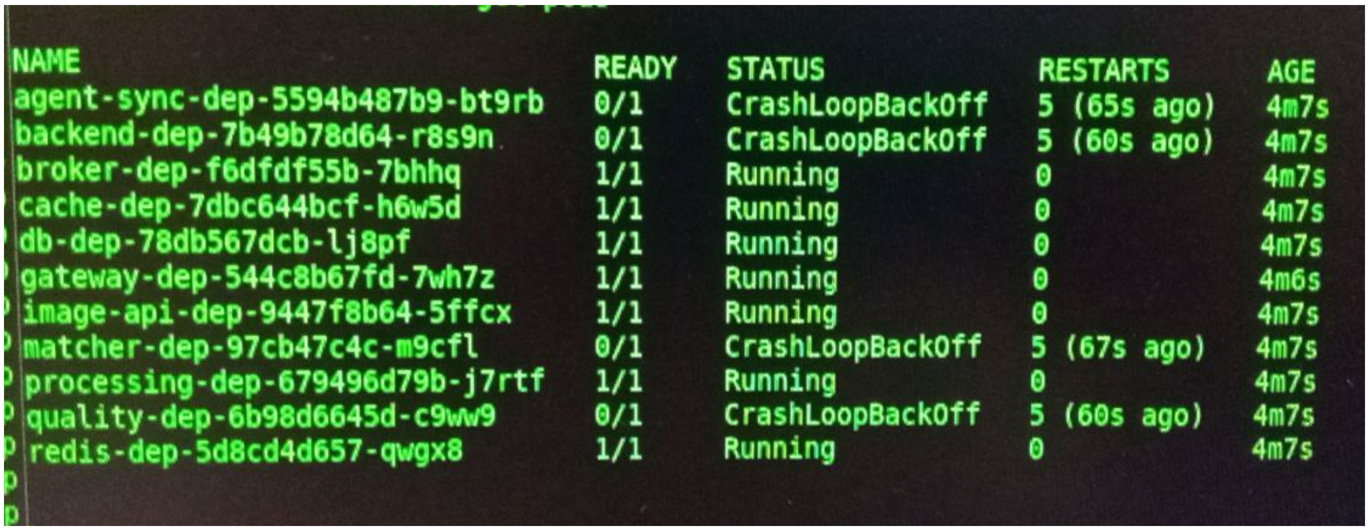
```
cat /proc/driver/nvidia/version
```



```
stranger@stranger:~/Platform/Platform_op-v1-9-1-rc3-fcb/on_premise$ cat /proc/driver/nvidia/version
NVRM version: NVIDIA UNIX x86_64 Kernel Module  515.65.01  Wed Jul 20 14:00:58 UTC 2022
GCC version:  gcc version 11.2.0 (Ubuntu 11.2.0-19ubuntu1)
```

Error occurred when deploying the Platform in a cluster

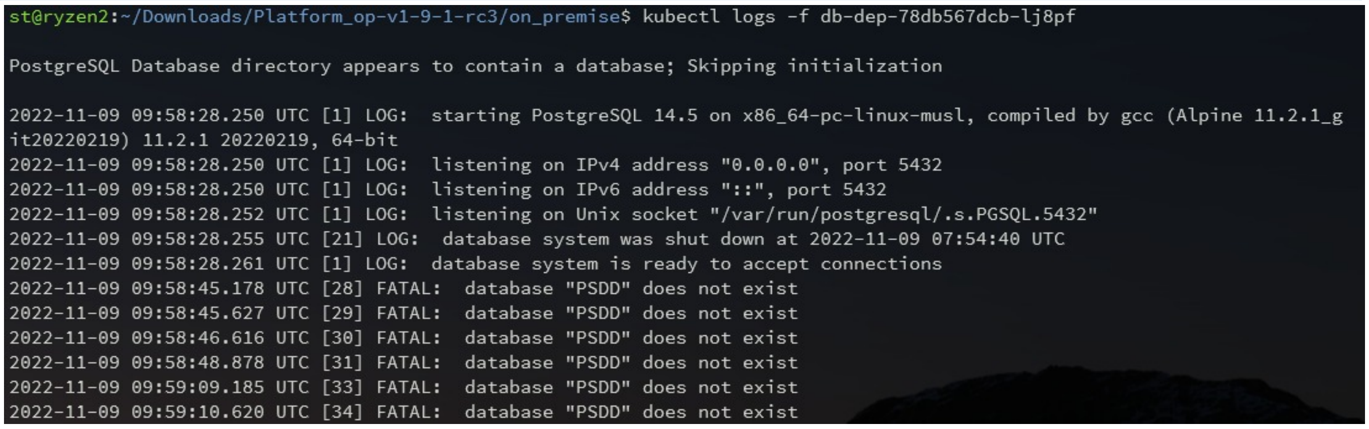
Problem: when executing a command `./setup/deploy.sh` you can get the following error:



```
NAME                                READY   STATUS             RESTARTS      AGE
agent-sync-dep-5594b487b9-bt9rb     0/1     CrashLoopBackOff   5 (65s ago)   4m7s
backend-dep-7b49b78d64-r8s9n        0/1     CrashLoopBackOff   5 (60s ago)   4m7s
broker-dep-f6dfdf55b-7bhhq          1/1     Running            0             4m7s
cache-dep-7dbc644bcf-h6w5d          1/1     Running            0             4m7s
db-dep-78db567dcb-lj8pf             1/1     Running            0             4m7s
gateway-dep-544c8b67fd-7wh7z        1/1     Running            0             4m6s
image-api-dep-9447f8b64-5ffcx       1/1     Running            0             4m7s
matcher-dep-97cb47c4c-m9cfl         0/1     CrashLoopBackOff   5 (67s ago)   4m7s
processing-dep-679496d79b-j7rtf     1/1     Running            0             4m7s
quality-dep-6b98d6645d-c9ww9        0/1     CrashLoopBackOff   5 (60s ago)   4m7s
redis-dep-5d8cd4d657-qwgx8          1/1     Running            0             4m7s
```

Solution: Request the log `db-dep` using the command:

```
kubectl logs -f <full name of pod>
```



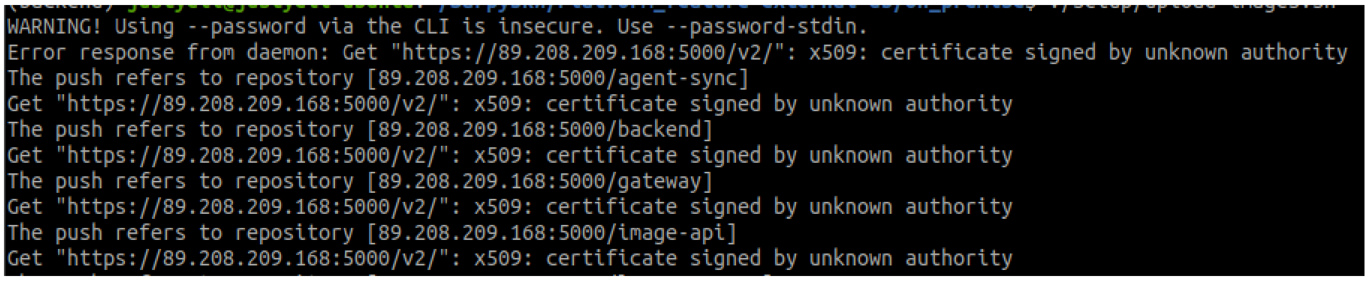
```
st@ryzen2:~/Downloads/Platform_op-v1-9-1-rc3/on_premise$ kubectl logs -f db-dep-78db567dcb-lj8pf

PostgreSQL Database directory appears to contain a database; Skipping initialization

2022-11-09 09:58:28.250 UTC [1] LOG:  starting PostgreSQL 14.5 on x86_64-pc-linux-musl, compiled by gcc (Alpine 11.2.1_g
it20220219) 11.2.1 20220219, 64-bit
2022-11-09 09:58:28.250 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
2022-11-09 09:58:28.250 UTC [1] LOG:  listening on IPv6 address "::", port 5432
2022-11-09 09:58:28.252 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2022-11-09 09:58:28.255 UTC [21] LOG:  database system was shut down at 2022-11-09 07:54:40 UTC
2022-11-09 09:58:28.261 UTC [1] LOG:  database system is ready to accept connections
2022-11-09 09:58:45.178 UTC [28] FATAL:  database "PSDD" does not exist
2022-11-09 09:58:45.627 UTC [29] FATAL:  database "PSDD" does not exist
2022-11-09 09:58:46.616 UTC [30] FATAL:  database "PSDD" does not exist
2022-11-09 09:58:48.878 UTC [31] FATAL:  database "PSDD" does not exist
2022-11-09 09:59:09.185 UTC [33] FATAL:  database "PSDD" does not exist
2022-11-09 09:59:10.620 UTC [34] FATAL:  database "PSDD" does not exist
```

If the output shows an error on incorrect database name or authorization data, redeploy the cluster (See the section 2.1).

Error occurred when uploading images to external registry

Problem: when uploading images you can get the following error:

```
The push refers to repository [<DOCKER_REGISTRY_SERVER>/<IMAGE>]
Get "<DOCKER_REGISTRY_SERVER>/v2/": x509: certificate signed by unknown authority
```



```
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Error response from daemon: Get "https://89.208.209.168:5000/v2/": x509: certificate signed by unknown authority
The push refers to repository [89.208.209.168:5000/agent-sync]
Get "https://89.208.209.168:5000/v2/": x509: certificate signed by unknown authority
The push refers to repository [89.208.209.168:5000/backend]
Get "https://89.208.209.168:5000/v2/": x509: certificate signed by unknown authority
The push refers to repository [89.208.209.168:5000/gateway]
Get "https://89.208.209.168:5000/v2/": x509: certificate signed by unknown authority
The push refers to repository [89.208.209.168:5000/image-api]
Get "https://89.208.209.168:5000/v2/": x509: certificate signed by unknown authority
```

Solution: Add or change the file `/etc/docker/daemon.json` and add your `DOCKER_REGISTRY_SERVER` to the list of insecure-registries:

```
{
    "insecure-registries" : [ "<DOCKER_REGISTRY_SERVER>" ]
}
```

Restart docker-service using the command below:

```
$ sudo systemctl restart docker
```

All the rest memory of the system where OMNI Platform is installed is cached

Problem: During installation, operation, and scaling of OMNI Platform, it may happen that all the remaining memory of the work system is cached.

Solution: In this case, OS manages the buffer/cache, not the platform. Cached memory is not used by other applications, but if necessary, OS will allocate it by reducing the buffer size.

Insufficient percentage of detected faces

Problem: The percentage of faces detected by the platform may not be sufficient for a specific use case.

Solution: In this case, you need to configure the parameters of the detector used (increase the detection threshold and the minimum and maximum sizes of the detected face).

To do this, stop the platform, open the `on_premise/deploy/values.yaml` file, and add the following line to the `env` block:

`FACE_SDK_PARAMETERS: "{\\"score_threshold\\": 0.7, \\"min_size\\": 150, \\"``max_size\\": 10000}"` , where:

`score_threshold` - detection threshold, from 0 to 1. The higher the threshold value, the more faces the detector can recognize;

`min_size` , `max_size` - minimum and maximum face size for detection, in pixels.