OneLab Enterprise

Installation & Update guide

Waters™ | Andrew Alliance™

Version	*Date*	*Author*	*Rationale*
1	07/07/2021	A. Pegaz-Blanc	First version
2	09/15/2021	A. Pegaz-Blanc	Update pre-requirements (Certificates and Intercom)
3	12/02/2021	A. Pegaz-Blanc	Add automatic update section and troubleshooting section

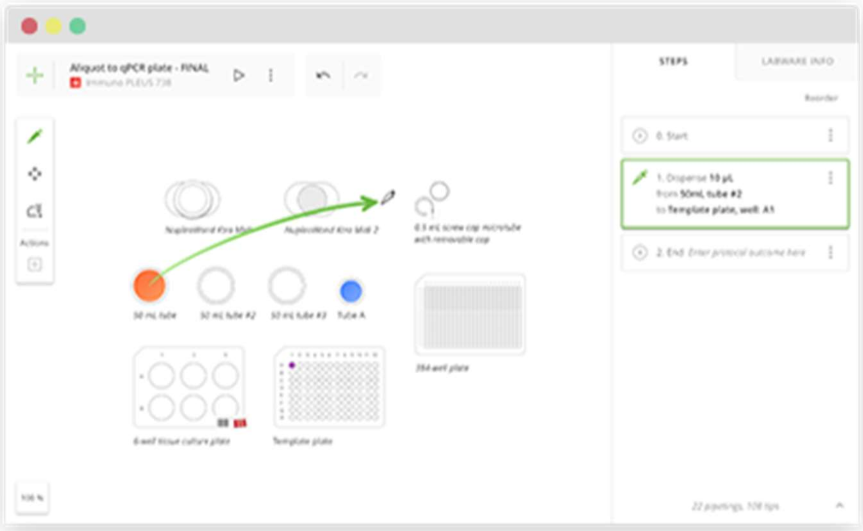
1. Presentation

1.1. OneLab

A unique browser-based software environment enabling researchers to design, and share, their own protocols, through a highly intuitive graphical interface that can then be executed step-by-step, from any PC or tablet.

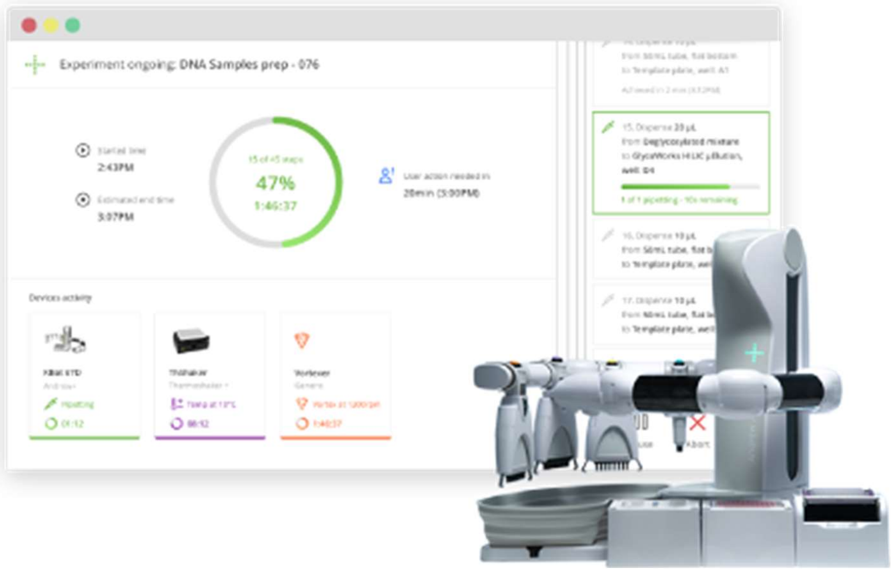
Design protocols

- Intuitive graphical drag-and- drop design interface.
- Eliminate error-prone calculations for serial dilutions and concentration normalizations.
- Accelerate collaboration and training by easily sharing protocols with other researchers.



Execute experiment

- Ensure correct manual execution of protocols, with your current set-up.
- Guarantee reproducibility with secure communication of protocols to OneLab compatible device(s).
- Connectivity with Andrew+ and Pipette+, or any other connected device, enabling researchers to achieve the highest levels of repeatability and productivity, with the added advantage of full traceability.



OneLab is free-to-use on onelab.andrewalliance.com but can also be deployed privately according to your needs (Enterprise or Standalone deployment)

More information on the website: https://www.andrewalliance.com/laboratory-software/ and on the Help Center: https://help.andrewalliance.com/

1.2. OneLab Enterprise

OneLab Enterprise is a private deployment on a server of your choice (private cloud, on your own network, etc.), providing the same features and possibilities as the public version, but also with the flexibility to:

- fully control and manage the server
- perform advanced integrations with LIMS and other external services
- control the software update according to your needs
- manage your own backups
- …

2. Pre-requirements

⚠ The pre-requirements have to be reviewed and validated prior to performing an installation or an update.

For any additional questions, please feel free to reach OneLab IT team through your sales representative or the support.

- **A non-root user with "sudo" privileges**

- **OneLab installation bundle**

Waters or Andrew Alliance should provide you the installation bundle "*onelab-enterprise-installer-X.X.X.zip*" (where X.X.X represents the version number)

- **OneLab server license (for installation only)**

Waters or Andrew Alliance should provide you a Server license.

This license (*license.onl* file) will be uploaded in OneLab through the administration interface.

- **Supported distributions by OneLab**
  - Ubuntu 16.04, 18.04 or 20.04
  - CentOS 7 or 8
  - RedHat 7 or 8
  - Debian 10
  - Fedora 33 or 34

- **Minimal technical requirements**
  - Min. 8 GB of RAM (depends on the # of users and the scalability policy)
  - Min. 64 GB of hard drive space (depends on the # of users and the scalability policy)

- **Installed dependencies/tools**
  - Docker CE (min. 20.10.0) & docker-compose
    - https://docs.docker.com/engine/install/ubuntu/
  - Ansible (min. 2.9.6)
    - Installing Ansible — Ansible Documentation
  - Libraries/Packages: “openssl”, “tar” “unzip” and “wget”

These requirements could also be installed during the OneLab installation.
The installation scripts are included in the installation bundle (in the *“/scripts”* folder).

- **SSL certificate and DNS configuration**
  - Public or Internal DNS domain (ex: onelab.mydomain.com)

Only a server/network configuration. It will allow the users to access OneLab via this URL.

- SSL certificates related to this domain (to enable the HTTPS)
  - server.key (private key)
  - server.pem (a single PEM file with the end entity certificate and the intermediate certificate)
  - chain.pem (a single PEM file with the intermediate certificate and the root certificate)

These files must be located on the server and will be used during the installation process.

- **External services configuration (optional)**
  - SMTP server access

SMTP access such as (credentials, URL, etc.)
Recommended if you want to use the experiment notifications feature.

- reCAPTCHA v2 account (https://developers.google.com/recaptcha/)

Recommended only if you want to enable external registration and if the server is publicly accessible.

- Intercom (https://www.intercom.com/)

Recommended only if you want to enable the support live-messaging system.
By default, Intercom will be enabled but you must allow Intercom’s communications as described in the following section.

- **Ports and data exchange**

OneLab instance must be reachable by the connected devices and by the user's computers according to the following requirements:

  - Between OneLab server and User's computers

Service	Source	Destination	Port	Type	Protocol	Direction
OneLab	User's PC	Private OneLab server DNS/IP	443	HTTPS	TCP	Outbound
HelpCenter (optional)	User's PC	help.andrewalliance.com (52.72.22.49)	443	HTTPS	TCP	Outbound
reCAPTCHA (optional)	User's PC	*.gstatic.com	443	HTTPS	TCP	Outbound
	User's PC	*.recaptcha.net	443	HTTPS	TCP	Outbound
Intercom (optional)	User's PC	*.intercom.io	443	HTTPS	TCP	Outbound
	User's PC	*.intercomcdn.com	443	HTTPS	TCP	Outbound
	User's PC	static.intercomassets.com	443	HTTPS	TCP	Outbound

  - Between OneLab server and connected devices:

Service	Source	Destination	Port	Type	Protocol	Direction
OneLab	Device	Private OneLab server DNS/IP	5671	AMQP	TCP	Bidirectional*
	Device	Private OneLab server DNS/IP	443	HTTPS	TCP	Outbound

** The data exchange is bidirectional (inbound and outbound traffics) but the communication with OneLab server is always started by the device (outbound connection).*

OneLab server should also be allowed to communicate with the external OneLab repository to get the latest versions and resources in case of update.

The restriction could only be allowed for the updates.

  - Between OneLab server and Internet:

Service	Source	Destination	Port	Type	Protocol	Direction
OneLab Repository	OneLab	hub.andrewalliance.com	443	HTTPS	TCP	Outbound
Docker Repository	OneLab	hub.docker.com	443	HTTPS	TCP	Outbound

3. Installation

We assume that the user is connected to the server with enough permission to install additional programs, and all the pre-requirements are done or correct.

1. Upload the installation bundle

The installation bundle *"onelab-enterprise-installer-X.X.X.zip"* must be uploaded on the server and located in the folder you want to install OneLab.

A "*/onelab*" folder will automatically be created in the next steps.

2. Unzip the installation bundle

Unzip the installation bundle into *"/onelab-enterprise-installer-X.X.X"* folder.

```
unzip onelab-enterprise-installer-X.X.X.zip -d onelab-enterprise-installer-X.X.X
```

3. Pre-requirement installation (optional)

If Ansible is NOT already installed, you can execute the following script to install it:

```
bash onelab-enterprise-installer-X.X.X/resources/scripts/distribution
/install_ansible.sh
```

Note: According to your distribution, please replace "distribution" by "centos", "debian", "ubuntu" or "redhat"

And, if Docker is NOT already installed, you can execute the following script to install it:

```
bash onelab-enterprise-installer-X.X.X/resources/scripts/distribution
/install_docker.sh
```

Note: According to your distribution, please replace "distribution" by "centos", "debian", "ubuntu" or "redhat"

4. Initialization

The installation will create a *"/onelab"* folder in your current folder.

Launch the installation script.

```
cd onelab-enterprise-installer-X.X.X
bash install.sh
```

Note: Do NOT execute the bash script directly from the current folder

Then, go back to the parent folder.

```
cd ..
```

During the initialization, by default:

A. A unique password database has been generated
B. A unique monitoring token has been generated (see "Monitoring" section)
C. A unique authentication token has been generated (see "Configuration" section)
D. A unique service authentication token has been generated (see "Configuration" section)

5. SSL configuration

Move your SSL files into *"/onelab/ssl"* folder respecting the file naming:

- server.key
- server.pem
- chain.pem

6. Edit configuration file

Edit the configuration file *"/onelab/configurations.yml"* (YAML file) with your favorite text editor.

```
nano onelab/configurations.yml
```

Note: nano is just for example

Edit the param *"onelab.domain"* and set your DNS domain (replacing *"https://localhost")* like "https://mydomain.com" (no "/" at the end).

The default settings are:

E. The external registration is disabled
F. The outgoing emails are disabled
G. The passwords will expire every 90 days
H. The min length for the passwords is 8 characters (including at least 1 digit, 1 upper case and 1 lower case character)
I. The user session will be logged out after 45min of idle time
J. Intercom (live support messaging system) is enabled

Please, refer to the "Configuration" section for more details.

7. Apply new configuration

Once the configuration is done, execute the reconfigure script to apply this configuration

```
cd onelab
bash reconfigure.sh
```

The reconfiguration will update applicative and internal files with the new configuration and will download the Docker images.

If for any reason, you have a network issue interrupting the downloading, you can re-do this task multiple times if needed until it is completed.

8. Start OneLab

Launch the following script to start OneLab

```
bash start.sh
```

OneLab will deploy the services one by one and ensure the deployment with a monitoring (see “Monitoring” section).

Once the script is successfully completed, OneLab can be considered as installed and operational.

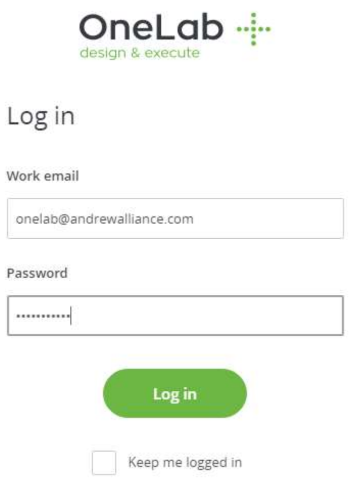
The first installation could take up to 5min the time to correctly bootstrap the database

9. First login and license upload

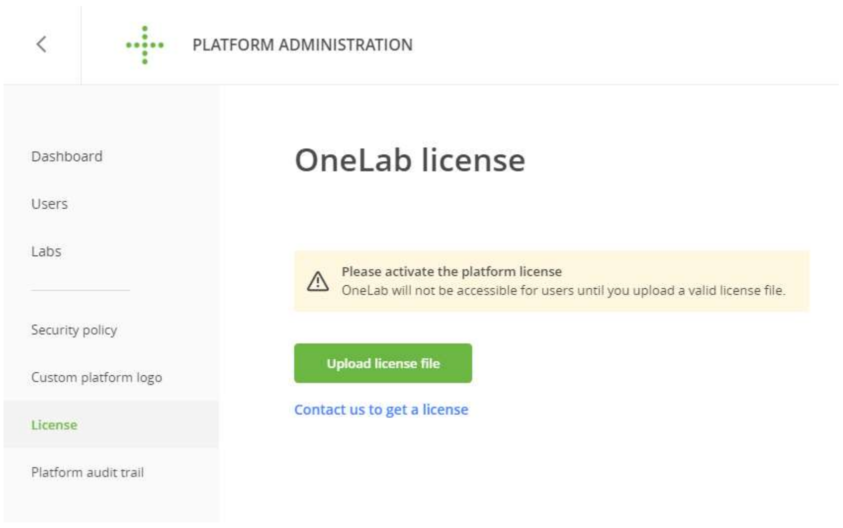
You should now be able to login on OneLab.

Open a browser and go to the domain you have configured to access the login page

- Username: onelab@andrewalliance.com
- Password: onelab@1Lab


OneLab
design & execute
Log in
Work email
onelab@andrewalliance.com
Password
•••••••••••

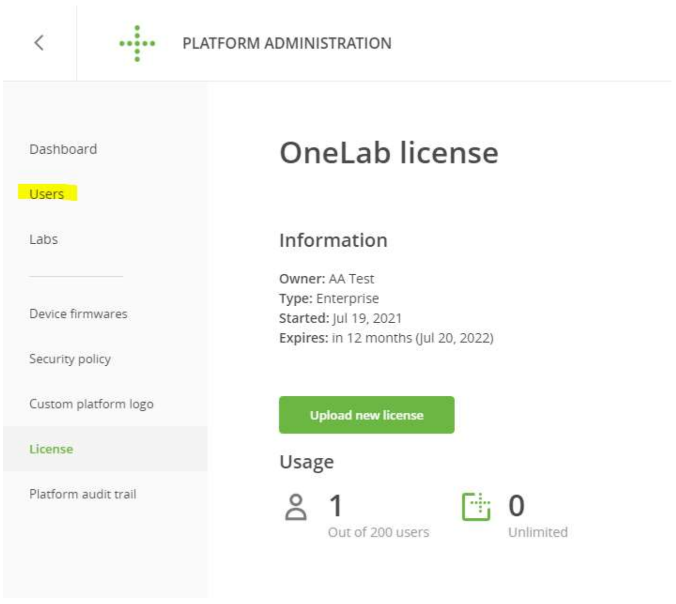
Then, you should be redirected to a page where you must upload the license file ("*license.onl*")



Please activate the platform license
OneLab will not be accessible for users until you upload a valid license file.

Once the license is uploaded, congratulations, the installation is done.

You could now redefine the admin account (email and password) in the "User" section of the administration and start using OneLab.



Information

Owner: AA Test
Type: Enterprise
Started: Jul 19, 2021
Expires: in 12 months (Jul 20, 2022)

Please check, the Help Center for more details: https://help.andrewalliance.com/en/

4. Automatic update to the latest version

The automatic update is available only if the current installed version is 1.13 or more.

If your current OneLab version is prior to 1.13.0, please refer to the next section "Manual update" to update your OneLab.

We assume that the user is connected to the server with enough permission to install additional programs, and all the pre-requirements are done or correct.

Go into the "*/onelab*" folder and execute the "*update.sh*" script

```
cd onelab
bash update.sh
```

The latest version will automatically be downloaded and installed on the current existing version.

A backup will be created in the "*/backups*" folder.

If OneLab is running, OneLab will be automatically stopped, and then restarted at the end of the update process.

5. Manual update to a specific version

We assume that the user is connected to the server with enough permission to install additional programs, and all the pre-requirements are done or correct.

1. Create a backup

We recommend performing a backup on OneLab before doing the update.

Please, refer to the "Backup" section.

2. Stop OneLab

OneLab must be stopped to perform the update.

```
cd onelab
bash stop.sh
```

Then, go back to the parent folder

```
cd ..
```

3. Upload the installation bundle

The installation bundle *"onelab-enterprise-installer-X.X.X.zip"* of the targeted X.X.X version must be uploaded on the server and located in the folder containing OneLab installation folder "*/onelab*".

```
./onelab-enterprise-installer-X.X.X.zip
./onelab
```

4. Unzip the installation bundle

Unzip the installation bundle into *"/onelab-enterprise-installer-X.X.X"* folder.

```
unzip onelab-enterprise-installer-X.X.X.zip -d onelab-enterprise-installer-X.X.X
```

5. Update OneLab

Launch the update script.

```
cd onelab-enterprise-installer-X.X.X
bash install.sh
```

Note: Do NOT execute the bash script directly from the current folder.

Then, go back to the parent folder.

```
cd ..
```

6. Re-apply configuration

Execute the reconfigure script to apply the existing configuration to the new files

```
cd onelab
bash reconfigure.sh
```

The reconfiguration will update applicative and internal files with the new configuration and will download the Docker images.

If for any reason, you have a network issue interrupting the downloading, you can re-do this task multiple times if needed until it is completed.

7. Start OneLab

Launch the following script to start OneLab

```
bash start.sh
```

OneLab will deploy the services one by one and ensure the deployment with a monitoring (see “Monitoring” section).

Once the script is successfully completed, OneLab can be considered as updated and operational.

6. Routines

6.1. Start OneLab

Execute the *"start.sh"* script

```
cd onelab
bash start.sh
```

6.2. Stop OneLab

Execute the *"stop.sh"* script

```
cd onelab
bash stop.sh
```

6.3. Backup OneLab

Execute the *"backup.sh"* script

```
cd onelab
bash backup.sh
```

The backup will create a new folder in *"/backups"* with the current date/time and the current OneLab version. In this folder, you will find 4 files:

- restore.sh : Script to call to restore this backup
- restore.yml : Ansible playbook called by the script
- data.tar.gz : Archive of the database
- onelab.tar.gz: Archive of the assets and configuration files

If you want to store the backup in another place, it is possible; you just must move and keep the 4 files together.

6.4. Restore a backup

⚠ The restore process will erase the current installation and the current data

The backup folder must be placed inside the *"/onelab/backups"* folder of an existing OneLab installation

You should have the following folders/files:

```
./onelab
  ...
  ./backups
    ./YYYYMMDDhhmmss_X.X.X/
      ./restore.sh
      ./restore.yml
      ./data.tar.gz
      ./onelab.tar.gz
```

1. Stop OneLab

OneLab must be stopped to perform the restore.

```
cd onelab
bash stop.sh
```

2. Restore the backup "YYYYMMDDhhmmss_X.X.X"

```
cd onelab/backups/YYYYMMDDhhmmss_X.X.X
bash restore.sh
```

The restore script will:

- Restore the assets
- Restore the configuration files and applicative files
- Restore the database

3. Re-apply configuration

Execute the reconfigure script to apply the existing configuration to the restored files

```
cd ../..
bash reconfigure.sh
```

4. Start OneLab

```
bash start.sh
```

6.5. Uninstall OneLab

⚠ Important: The uninstall process will erase the current installation, the current data, and all the backups stored in /onelab/backups

Please, perform a backup and/or move the existing backups to another location if you want to keep the data.

1. Execute the uninstall script

```
cd onelab
bash installation/latest/scripts/uninstall.sh
```

2. **Remove OneLab folder**

```
cd ./..
rm -rf onelab
```

6.6. Monitoring

When OneLab is running, it is possible to monitor it through the following HTTP call:

- https://**mydomain.com**/api/status?token=**MonitoringToken**

Replacing "*mydomain.com*" by your domain, and the monitoring token by the token defined in the configuration file (*"/onelab/configurations.yml"*) at "*onelab.security.monitoring.token".*

The HTTP call (GET) will return a JSON payload (code 200) with the following information:

- version: version of OneLab
- status: status of OneLab between
  - running: (all the services are running)
  - degraded: (all the services are running, but at least one service is experiencing trouble)
  - not-running: (at least one service is not running at all)
- upTime: up-time of OneLab since the last reboot
- startDate: date of the last reboot
- services: list of the services with detailed info

In case of different HTTP codes:

- HTTP Code 403: The token is not correct
- HTTP Code 404: API is not running, OneLab is probably not running
- HTTP Code 503: The service is unavailable. OneLab is either still booting, or either not running

7. Configuration

The OneLab configuration file (*"configurations.yml"*) is a YAML file located at the root level of the installation folder (*"/onelab"*).

The YAML format (http://yaml.org/) is a strict format where the indentation (2 spaces) is important.

Time format parameters are defined with a value and a unit, like:

- "10s" for 10 seconds
- "5m" for 5 minutes
- "30h" for 30 hours
- "2d" for 2 days

It is also possible to comment a line by starting it with # character.

After a configuration change, you will have to apply the new configuration:

```
cd onelab
bash reconfigure.sh
```

And restart OneLab.

```
cd onelab
bash stop.sh
bash start.sh
```

The configuration is composed of multiple sections having one or more parameters.

7.1. Domain

The section has only one parameter *“domain”*.

domain (string)

URL of OneLab. Used to configure the HTTP server and to generate correctly access links (ex: https://onelab.mydomain.com) Without any “/” at the end.

Example

```
onelab:
  domain: https://onelab.mydomain.com
  …
```

7.2. Logs

logs.path *(string)*

Path to the log folder. OneLab log files will be generated there.

logs.level *(string)*

Log level (between error, warn, info, verbose, debug)
verbose and debug are not recommended for production environment

Example

```
onelab:
  …
  logs:
    path: "./logs"
    level: info
  …
```

7.3. Assets

assets.path *(string)*

Path to the assets folder. OneLab will store all the assets in this folder (thumbnails, user avatar, device logs, etc.).

assets.purge *(time format)*

Frequency of the purge, it will remove what can be removed.

Example

```
onelab:
  ...
  assets:
    path: ". /data"
    purge: 1d
  ...
```

7.4. Security

security.cors (string or list)

CORS (Cross-origin resource sharing) configuration for the incoming call to OneLab API.

By default, “*” value accepts all the origins.

You can list specific origin If needed, like:

```
security:
  cors:
    •   mydomain.com
    •   mydomain.org
```

security.auth.token

Authentication token configuration.

2 parameters:

- **expiration** *(time format)*
  Define the lifetime of an auth token. The token is automatically refreshed by the app. Having a short duration is better and prevents from stolen token threat.
- **key** *(string)*
  Token used as Hash/Checksum to validate the session on server side.
  Changing this token will invalidate all the current auth tokens (by blocking the refresh and the validation)
  The users will be forced to login to renew their tokens

security.auth.password

Password configuration management.

2 parameters:

- **expiration** *(time format)*
  Define the lifetime of a password. The user will be forced to redefine a password if the password is expired.
- **min_length** *(number)*
  Number of minimum characters required by the password policy.
  The password policy will also require at least 1 lower case character, 1 upper case character and 1 digit. Not configurable.

security.auth.allow_list *(string or list)*

Define the list of the allowed email domains.

Only the listed domains will be allowed to login and register (if registration is enabled)

If no domain is listed (*"[]"* value), there is no restriction, and all the domains are allowed.

security.auth.block_list *(string or list)*

Define the list of the blocked email domains.

The listed domains will not be allowed to login and register (if registration is enabled)

If no domain is listed (*"[]"* value), there is no restriction, and all the domains are allowed.

security.ratelimit.ip

IP rate limiter security configuration according to the IP of the incoming calls preventing spam and abusive uses.

2 parameters:

- **max** *(number)*
  Number of calls before blocking the next incoming calls from the IP on the rolling period
- **duration** *(time format)*
  Duration of the rolling period allowing a max number of calls.

security.ratelimit.auth

Authentication rate limiter security configuration preventing abusive uses of login form/calls.

4 parameters:

- **max** *(number)*
  Number of login attempts before blocking the email according to the duration param
- **duration** *(time format)*
  Block duration. The user will not be able to retry to login during this period.
- **delay_after** *(number)*
  Number of login attempts before adding a delay between the attempts
- **delay_ms** *(number)*
  Duration of the delay in milliseconds between 2 attempts when the number of "*delay_after*" is reached

security.device.cors *(string or list)*

CORS (Cross-origin resource sharing) configuration for the incoming call to OneLab Device API. This API is used to pair and synchronize the connected devices.

By default, "*" value accepts all the origins.

You can list specific origin If needed, like:

security:
  devices:
    cors:
- *mydomain.com*
- *mydomain.org*

security.monitoring.token *(string)*

Token use to access to the monitoring URL:
https://**mydomain.com**/api/status?token=**MonitoringToken**

See the Monitoring section for more details.

Example

```
onelab:
  ...
  security:
    cors: '*'
    auth:
      token:
        expiration: 5m
        key: MyAuthToken
      password:
        expiration: 90d
        min_length: 8
      allow_list:
        - mydomain.com
        - mydomain.org
      block_list: []
    ratelimit:
      ip:
        max: 1000
        duration: 1d
      auth:
        max: 5
        duration: 5m
        delay_after: 2
        delay_ms: 1000
    devices:
        cors: '*'
    monitoring:
        token: MyMonitoringToken
  ...
```

7.5. Params

params.session.idle *(time format)*

Delay before considering the user as idle and close his session

params.session.remember_me *(boolean)*

Enable or not the “Remember Me” feature on the login

params.lab.creation_policy *(string)*

Define the Lab creation policy, it could be:

- *none*
  Users are not allowed to create a new Lab. Platform admin will have to do it.
- *one*
  Users can create a new Lab only if they have no Lab.
- *many*
  Users can create as many Labs as they want

params.signup *(boolean)*

Enable or not the external registration.

If disabled, the user will not be able to register by himself.

params.ga *(string)*

Define and enable Google Analytics tracker.

params.intercom

Define and enable Intercom live-messaging system.

2 parameters:

- **appid** *(string)*
  Intercom AppID
- **secret** *(string)*
  Intercom secret token

params.recaptcha

Define and enable reCaptcha v2 service.

2 parameters:

- **client** *(string)*
  reCaptcha client ID
- **secret** *(string)*
  reCaptcha secret token

Example

```
onelab:
  ...
params:
    session:
      idle: 45m
      remember_me: true
    lab:
      creation_policy: many
    signup: false
    # Google Analytics
    # ga: XXXXX
    # Intercom Andrew Alliance
    # intercom:
    #  appid: XXXXXXX
    #  secret: XXXXXX
    # Recaptcha Andrew Alliance
    # recaptcha:
    #  client: XXXXXXXXXXXI
    #  secret: XXXXXXXXXXXI
  ...
```

7.6. Mailer

mailer.smtp *(optional)*

Define and enable the standard SMTP connector

- **host** *(string)*
  Hostname of the SMTP server
- **port** *(number)*
  Port of the SMTP server
- **auth** *(optional)*
    - **user** *(string) (optional)*
      Username
    - **pass** *(string) (optional)*
      Password for the user if the normal login is used
    - **type** *(string) (optional)*
      Authentication type, defaults to 'login', other option is "oauth2"
    - **method** *(string) (optional)*
      defines preferred authentication method, defaults to "PLAIN"
- **secure** *(boolean)*
  Define if the connection must use TLS or not

For more details and advanced parameters, please check https://nodemailer.com/smtp/

mailer.ses (optional)

Define and enable the AWS SES connector (https://aws.amazon.com/ses/)

- **accessKeyId** *(string)*
  AWS IAM Access Key ID
- **secretAccessKey** *(string)*
  AWS IAM Secret key

mailer.mailgun *(optional)*

Define and enable the Mailgun connector (https://www.mailgun.com/)

- **auth**
    - **api** *(string)*
      API key
    - **domain** *(string)*
      Domain secret token

mailer.debug *(optional)*

Debug configuration for the mailer.

- **type** *(string)*
  Type of the debug "*mail*" or "*file*"
  mail value is associated with the *redirect* param. All the outgoing emails will be redirected to the email address defined in redirect.
  file value is associated with the *path* param. All the outgoing emails will be saved in the folder defined by *path* value
- **path** *(string) (optional)*
  Destination folder for the outgoing emails if *file* debug is enabled
- **redirect** *(string) (optional)*
  Redirection email if *mail* debug is enabled

Not recommended for production environment, but could be useful for pre-test or pre-prod environments

mailer.noreply *(string)*

Define the no-reply email address used as sender by OneLab

mailer.queue

Define and configure the mailing task.

- **scheduling** *(number)*
  Number of seconds before checking if there are new mails to send
- **maxsize** *(number)*
  Max number of emails sent in parallel

mailer.error

Define and configure how to handle the SMTP error

- **maxtries** *(number)*
  Number of sending tries before considering an email in error
- **timeout** *(number)*
  Timeout in seconds before considering a sent email in error

Example

```
onelab:
 ...
 mailer:
   #
   # SMTP
   # smtp:
   #   host: XXX
   #   port: XXX
   #   auth:
   #     user: XXX
   #     pass: XXX
   #     type: custom | login | oauth2
   #     method: XXX
   #   secure: true | false
   #
   # Amazon SES
   # ses:
   #   accessKeyId: XXXX
   #   secretAccessKey: XXXX
   #
   # MailGun
   # mailgun:
   #   auth:
   #     api: XXXX
   #     domain: XXXX
   #
   # Debug
   # debug:
   #   type: file | mail
   #   path: XXXX
   #   redirect: XXX@andrewalliance.com
   #
   #noreply: no-reply@andrewalliance.com
   queue:
     scheduling: 15
     maxsize: 50
   error:
     maxtries: 3
     timeout: 60
 ...
```

7.7. Services

services.db

Configuration of the Database.

- **host** *(string) (optional)*
  Hostname, *db* by default (referencing the Docker db service)
- **port** *(number) (optional)*
  Port of the database. 5432 by default
- **database** *(string)*
  database name of the database
- **username** *(string)*
  username of the database
- **password** *(string)*
  password of the database
- **schema** *(string)*
  Schema name of the database
- **replicas** *(number) (optional)*
  Number of instances of db service in OneLab, 1 by default.

services.redis

Configuration of Redis

- **host** *(string) (optional)*
  Hostname, *redis* by default (referencing the Docker Redis service)
- **port** *(number) (optional)*
  Port. 6379 by default
- **replicas** *(number) (optional)*
  Number of instances of Redis service in OneLab, 1 by default.

services.rabbit

Configuration of RabbitMQ

- **url** *(string) (optional)*
  Hostname, *rabbitmq* by default (referencing the Docker RabbitMQ service)
- **port** *(number) (optional)*
  Port. 5671 by default
- **token** *(string)*
  Internal token used by OneLab services to access RabbitMQ Interval VHost
- **replicas** *(number) (optional)*
  Number of instances of RabbitMQ service in OneLab, 1 by default.

services.*

Configuration for the other services:

- api: OneLab API
- apidevice: Device API (for pairing and first synchronization with devices)
- apirabbit: Internal Rabbit MQ API for RabbitMQ authentication
- devices: Worker managing the device commands and requests
- experiments: Worker managing the ongoing experiments
- images: Worker generating the Protocol thumbnails
- mailer: Worker handling the outgoing emails
- manual: Worker handling the Manual simulations and executions
- ws: Websocket worker handling the Websocket connections with the front apps

All the services are configurable in the same way:

- **replicas** *(number) (optional)*
  Number of instances of the service in OneLab, 1 by default.

If the number of simultaneous connected users and/or connected devices are less than 100, we recommend only scale the *api* service to 2 or 3 and leave the other service at 1.

Example

```
onelab:
  ...
  services:
    db:
      # host: db
      # port: 5432
      database: postgres
      username: postgres
      password: DBPasswordPlaceholder
      schema: onelab
      # replicas: 1
    # redis:
    #   host: redis
    #   port: 6379
    #   replicas: 1
    rabbit:
      # url: rabbitmq
      # port: 5671
      token: TokenRabbitPlaceholder
      # replicas: 1
    api:
      replicas: 2
    # apidevice:
    #   replicas: 1
    # apirabbit:
    #   replicas: 1
    # devices:
    #   replicas: 1
    # experiments:
    #   replicas: 1
    # images:
    #   replicas: 1
    # mailer:
    #   replicas: 1
    # manual:
    #   replicas: 1
    # ws:
    #   replicas: 1
 ...
```

8. Troubleshooting

8.1. Docker repository connection

Impossible to connect to the Docker repository (flagged as Insecure)

Due to the network/infrastructure configuration, the repository hub.andrewalliance.com could be seen as insecure and will reject the login and/or the pull requests.

Create a new file at “/etc/docker/daemon.json”

```
nano /etc/docker/daemon.json
```

And put the following content:

```
{"insecure-registries": ["hub.andrewalliance.com"]}
```

Then, restart Docker

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

8.2. Firmware update issue

Connected device is not able to update its firmware

If the connected device is not able to perform the auto-update of the firmware, it’s probably to a HTTPS communication error between the device and OneLab server.

You should:

- Check if OneLab domain is well configured in the configurations.yml file (See section 6.1)

- Check if the device is able to resolve and ping the OneLab domain (especially if the device is on a different network)

If the problem persists, feel free to contact the Support via Intercom or by mail.