Creatio Academy

Application server on Linux

Version 7.17

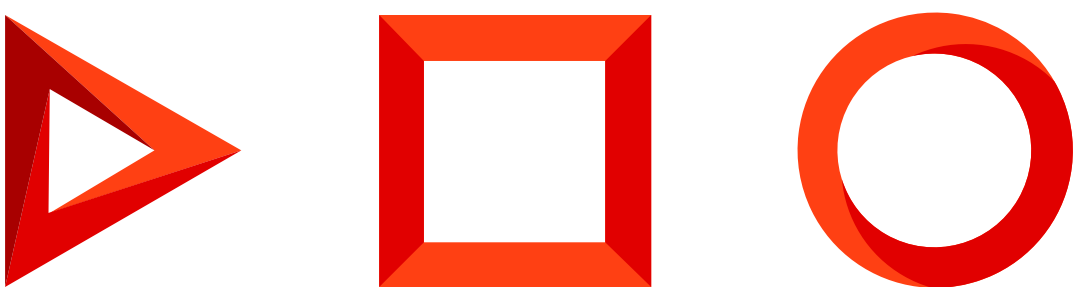

Table of Contents

Prepare to deploy Creatio .NET Core on Linux

PRODUCTS: ALL CREATIO PRODUCTS

Attention. Support for .NET Core 3.1 will be retired in Creatio version 8.0.9. We recommend moving to .NET 6 when updating Creatio to version 8.0.8.

Prepare the setup files

Creatio setup files come as a zip archive. To unpack the archive, run:

```
unzip -d /path/to/application/directory/ CREATIO_ARCHIVE_NAME.zip
```

where **/path/to/application/directory/** is the directory to which to extract the files. We recommend unpacking the archive to the current directory:

```
unzip creatio_archive_name.zip
```

Where **creatio_archive_name.zip** is the name of the Creatio setup archive.

Note. If running the unzip command results in the “command not found” error, install the *.zip file de-archiver:

```
sudo apt-get install unzip
```

Set up ConnectionStrings.config

The ConnectionStrings.config file in the Creatio root directory stores the connection parameters of the database and external services for your application.

Edit the ConnectionStrings.config file

1. Go to the root directory of the Creatio application **~\WebAppRoot\Creatio**.
2. Open the ConnectionStrings.config file in a text editor.

3. Specify the **connectionStrings** connection parameters of your site.

A sample ConnectionStrings.config file.

```
<?xml version="1.0" encoding="utf-8"?>
<connectionStrings>
    <add name="db" connectionString="Server=[Database server name];Port=31436;Database=[Database
    <add name="redis" connectionString="host=[Machine name];db=[Redis DB number];port=6379" />
    <add name="tempDirectoryPath" connectionString="%TEMP%/%USER%/%APPLICATION%" />
    <add name="influx" connectionString="url=[Address of the analytics collection service]; user
    <add name="clientPerformanceLoggerServiceUri" connectionString="[Logging service address]" /
    <add name="messageBroker" connectionString="amqp://[MessageBroker username]:[Password]@[Addr
</connectionStrings>
```

Required ConnectionStrings.config settings

Creatio requires the database and caching server connection parameters for operation.

- **db** manages the database connection. Configure the path to the database to connect and the authorization method on the database server in this parameter.

```
<add name="db" connectionString="Server=[ Database server name ];Port=31436;Database=[ Databa
```

- **redis** manages the interaction with the Redis server:

```
<add name="redis" connectionString="host=[ Machine name ];db=[ Redis DB number ];port=6379; "
```

Attention. The Redis database number must be unique for each Creatio site.

Optional ConnectionStrings.config settings

The external service connection parameters are optional. Fill them out only if your Creatio configuration requires it. For example, do that if you want to collect the site analytics.

- **tempDirectoryPath** is the path to the temporary directory the package installation mechanism requires:

```
<add name="tempDirectoryPath" connectionString=[ Path to the temporary directory the package
```

- **Influx** manages the interaction with the site analytics collection service. Fill out this parameter only if you need to collect the functionality use analytics for debugging.

```
<add name="influx" connectionString="url=[ Site analytics collection service ]; user=[ User o
```

- **clientPerformanceLoggerServiceUri** manages the interaction with the logging service. Fill out this parameter only if you need to collect the data about how Creatio pages load.

```
<add name="clientPerformanceLoggerServiceUri" connectionString="[ Logging service address ]"
```

- **messageBroker** manages the interaction with the RabbitMQ service. Fill out this parameter only if you need to set up the horizontal load scaling using RabbitMQ.

```
<add name="messageBroker" connectionString="amqp://[ MessageBroker username ]:[ Password ]@[
```

Deploy the Creatio .NET Core application server on Linux

PRODUCTS: ALL CREATIO PRODUCTS

Attention. Support for .NET Core 3.1 will be retired in Creatio version 8.0.9. We recommend moving to .NET 6 when updating Creatio to version 8.0.8.

Before you deploy the server, take the following steps:

- Prepare the Creatio setup files. Read more >>>
- Deploy the database server. Read more >>>
- Deploy the Creatio caching server (Redis). Read more >>>
- Modify the ConnectionStrings.config file. Read more >>>

Note. Learn more about running a PostgreSQL server in Docker in the Docker documentation.

Method 1. Deploy Creatio .NET Core on Linux directly

To deploy the Creatio application server:

- Install .NET Core, a GDI+ compatible API for non-Windows operating systems, and development libraries and header files for GNU C. Read more >>>
- Run the Creatio application server. Read more >>>

Install .NET Core and other Creaito dependencies

1. Download the packages-microsoft-prod package:

```
wget -q https://packages.microsoft.com/config/ubuntu/18.04/packages-microsoft-prod.deb -O pac
```

2. Log in as root:

```
sudo su
```

3. Install the downloaded package:

```
dpkg -i packages-microsoft-prod.deb
```

4. Update the package lists:

```
apt-get update
```

5. Install the APT transport for downloading via the HTTP Secure protocol:

```
apt-get install apt-transport-https
```

6. Update the package lists:

```
apt-get update
```

7. Install .NET Core:

```
apt-get install dotnet-sdk-3.1
```

8. Install a GDI+ compatible API for non-Windows operating systems:

```
apt-get install -y libgdiplus
```

9. Install development libraries and header files for GNU C:

```
apt-get install -y libc6-dev
```

10. Log out from your root session:

```
exit
```

Run the Creatio application server

Note. If you are deploying the .NET Core development environment with access via HTTP, modify the Terrasoft.WebHost.dll.config file in the Creatio root directory before you run Creatio. Set the “add key” parameter to the following:

```
<add key="CookiesSameSiteMode" value="Lax" />
```

This ensures correct operation both via HTTP and HTTPS. However, the mobile app will not be operational if you use this setting.

To run Creatio:

1. Open the directory that contains Creatio setup files:

   ```
   cd /path/to/application/directory/
   ```

2. Run the .NET Core server:

   ```
   COMPlus_ThreadPool_ForceMinWorkerThreads=100 dotnet Terrasoft.WebHost.dll
   ```

Creatio HTTP version will be available on port 5000.

Creatio HTTPS version will be available on port 5002.

Note. To log in to newly deployed Creatio, use the default Supervisor user account. It is highly recommended to change the Supervisor password immediately. Login: Supervisor; password: Supervisor.

Method 2. Deploy Creatio .NET Core on Linux using Docker

Use this deployment method to run a compartmentalized Creatio application. We assume that you have installed the Redis sever, deployed the Creatio database, and set up the ConnectionStrings.config file using the instructions in the previous steps.

To deploy Creatio application server using Docker:

- Make Redis accessible from the Docker container. Read more >>>
- Install Docker. Read more >>>
- Create a Dockerfile. Read more >>>
- Build and run a Docker image. Read more >>>

Attention. We recommend deploying Creatio using Docker for development and testing environments only. Avoid using Docker for the production environment before we implement support for Creatio updates in Docker (planned for upcoming releases).

Configure the Creatio caching server (Redis)

1. Open **redis.conf** in a text editor as root. For example, use the Nano text editor:

```
sudo nano /etc/redis/redis.conf
```

2. Locate the "**bind 127.0.0.1 ::1**" entry. Replace the entry with "**bind 0.0.0.0**" to listen to all available IPV4 interfaces.
3. Save changes and exit the editor.
4. Restart the Redis server:

```
sudo systemctl restart redis-server
```

Install Docker

To install Docker, run:

```
sudo apt-get install docker
```

Create a Dockerfile

/path/to/application/directory/ is the directory that contains unpacked Creatio installation files.

1. Open the Creatio directory:

```
cd /path/to/application/directory/
```

2. Create a Dockerfile using a text editor. For example, use the Nano text editor:

```
nano Dockerfile
```

3. Insert the following code:

```
FROM mcr.microsoft.com/dotnet/core/sdk:3.1 AS base
EXPOSE 5000 5002
```

```
RUN apt-get update && \
apt-get -y --no-install-recommends install \
libgdiplus \
libc6-dev && \
apt-get clean all && \
rm -rf /var/lib/apt/lists/* /var/cache/apt/*
WORKDIR /app
COPY . ./
FROM base AS final
WORKDIR /app
ENV ASPNETCORE_ENVIRONMENT Development
ENV TZ US/Eastern
ENV COMPlus_ThreadPool_ForceMinWorkerThreads 100
ENTRYPOINT ["dotnet", "Terrasoft.WebHost.dll"]
```

4. Press Ctrl+O to save the changes.
5. Press Ctrl+X to exit the editor.

Build and run a Docker image

Note. If you are deploying the .NET Core development environment with access via HTTP, modify the Terrasoft.WebHost.dll.config file in the Creatio root directory before you run the Docker image. Set the “add key” parameter to the following:

```
<add key="CookiesSameSiteMode" value="Lax" />
```

This ensures correct operation both via HTTP and HTTPS. However, the mobile app will not be operational if you use this setting.

Build a Docker image:

```
docker build -f Dockerfile -t creatioimg .
```

Run the docker image:

```
docker run -p http_port_number:5000 -p https_port_number:5002  -d --dns=DNS_server_ip --dns-sear
```

http_port_number is a port number. Docker will serve the HTTP version on this port

https_port_number is a port number. Docker will serve the HTTPS version on this port

DNS_server_ip is the IP address of a DNS server that enables the container to resolve Internet domains. You can use multiple **--dns** flags for multiple DNS servers.

DNS_address_suffix is a DNS search domain that enables the container to search for non-fully-qualified hostnames. You can use multiple **--dns-search** flags for multiple DNS search domains.

> **Note.** Add the --restart=always flag to the command make a persistent Docker container.

Creatio HTTP version will be available on port **http_port_number**.

Creatio HTTPS version will be available on port **https_port_number**.

> **Note.** To log in to a newly deployed application, use the default Supervisor user account. It is highly recommended to change the Supervisor password immediately. Login: Supervisor; password: Supervisor.

Configure Creatio .NET Core for HTTPS

Before you start working in Creatio via HTTPS, take the following steps:

1. Obtain a *.pfx digital certificate from the certification center.

   > **Note.** If you use a self-signed certificate, Creatio mobile application cannot connect to the Creatio website due to security policies of mobile applications.

2. Go to Creatio root directory and open appsettings.json.
3. Specify your website address, path to the certificate, and certificate password in the “Https” block.

Example of the “Https” block in appsettings.json

```
"Https": {
    "Url": "https://::5002",
    "Certificate": {
    "Path": "C:\\Projects\\site\\20210215_103239\\localhost.pfx",
    "Password": "Password"
    }
}
```

> **Note.** You can specify both relative and absolute path to the certificate. The absolute path must be JSON compatible.