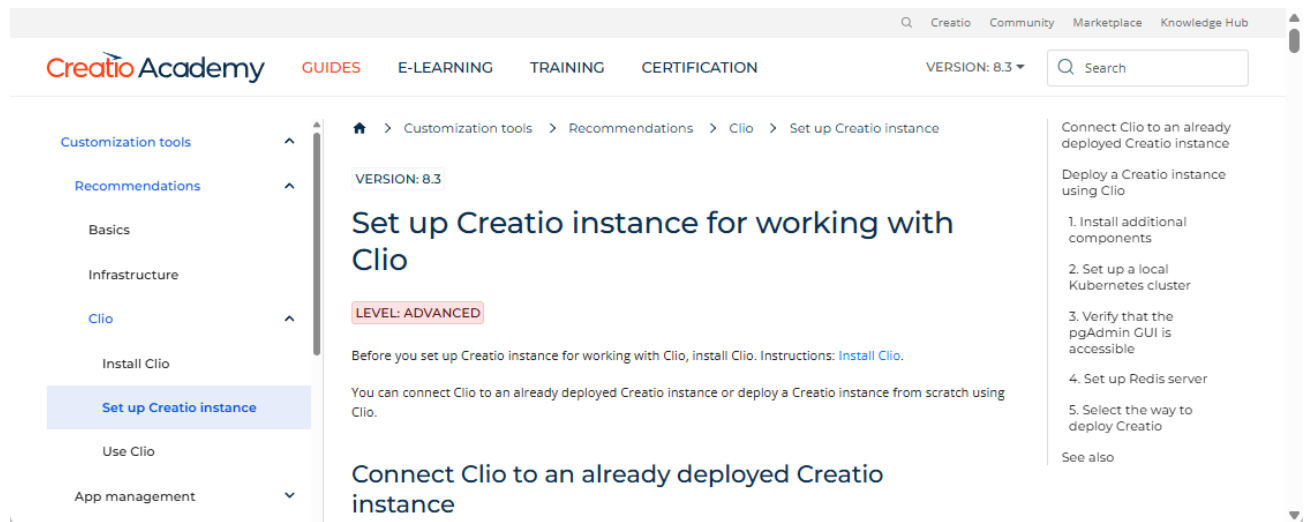


The current place of the article in the documentation structure:



VERSION: 8.3

# Set up Creatio instance for working with Clio

LEVEL: ADVANCED

Before you set up Creatio instance for working with Clio, install Clio. Instructions: [Install Clio](#).

You can connect Clio to an already deployed Creatio instance or deploy a Creatio instance from scratch using Clio.

## Connect Clio to an already deployed Creatio instance

1. **Open the terminal of Visual Studio Code** or **run the Windows PowerShell** as administrator.
2. **Run the** `clio reg-web-app SomeEnvironmentName -u SomeCreatioURL -l SomeLogin -p SomePassword` **command**, where:
  - `SomeEnvironmentName` is the name of your Creatio environment.

- `SomeCreatioURL` is the URL of already deployed Creatio instance.
- `SomeLogin` is the user login to the Creatio instance.
- `SomePassword` is the user password to the Creatio instance.

## Deploy a Creatio instance using Clio

Clio lets you deploy Creatio on .NET and .NET Framework platforms and use MS SQL and PostgreSQL databases.

### 1. Install additional components

1. **Install Linux on Windows** (for Windows only). Instructions: [How to install Linux on Windows with WSL](#) (official vendor documentation).

View the example that configures `.wlsconfig` file below.

`.wlsconfig` file

```
[wsl2]
memory=8GB # Limits VM memory in WSL 2 to 16 GB
processors=4 # Makes the WSL VM use 8 virtual processors
```

2. **Install Rancher Desktop** that creates a local Kubernetes cluster.  
Instructions: [Rancher Desktop by SUSE](#) (official vendor documentation).

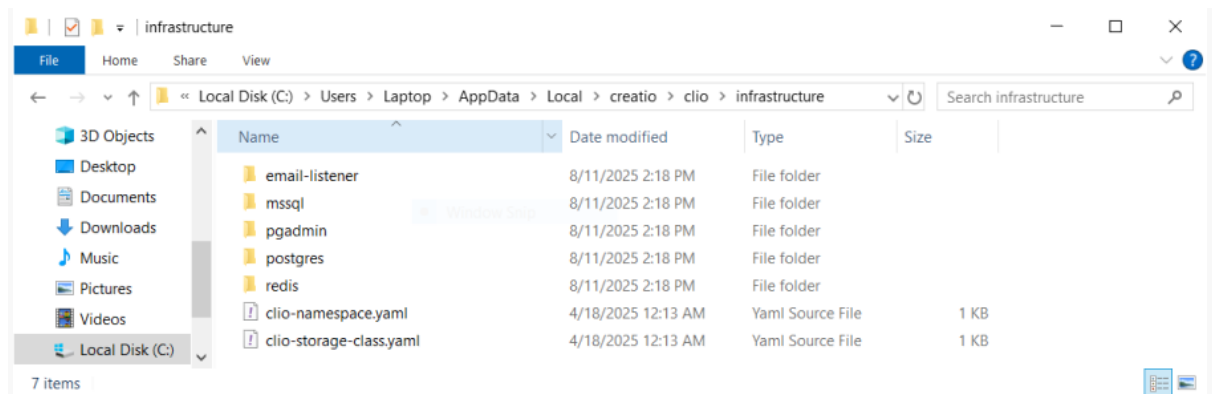
We recommend using the latest stable Kubernetes version and docker Container Engine.

3. **Make sure that you have required Windows components enabled** (for Windows only).
  - i. Run the `clio check-windows-features` command.
  - ii. Check enabled components. Enable required components if needed.  
Instructions: [Enable required Windows components](#).

### 2. Set up a local Kubernetes cluster

1. **Run the** `kubectl cluster-info` **command** to make sure that local Kubernetes cluster is installed.
2. **Run the** `clio create-k8-files` **command** to generate k8s manifests for required components. This creates the `C:\Users\SomeWindowsUser\AppData\Local\creatio\clio\infrastructure` directory (Fig. 1) that includes pre-configured services of your environment, such as Email Listener, MS SQL server, PostgreSQL server, Redis server and pgAdmin GUI to manage PostgreSQL databases.

Fig. 1 infrastructure directory



3. **Set up paths to the generated k8s manifests.**
  - i. Go to the `infrastructure` directory in Windows PowerShell.
  - ii. Run the following commands.

```
# Retrieve the path to the "appsettings.json" file and store it in
the "$appsettings" variable.
$appsettings = clio externalLink clio://GetAppSettingsFilePath

# Extract the directory path of the "appsettings.json" file and
store it in the "$dir" variable.
$dir = Get-Item $appsettings | Select-Object -ExpandProperty
DirectoryName

# Change the current location to the "infrastructure" directory
located next to "appsettings.json" file.
Set-Location -Path $dir/infrastructure

# Display all files and subdirectories in the "infrastructure"
directory.
Get-ChildItem -Path $dir/infrastructure
```

4. **Make sure your kubectl context is set to Rancher Desktop.** Otherwise, your Creatio instance will be deployed into another Kubernetes cluster.
5. **Modify nested files** in the `infrastructure` subdirectories to redefine CPU resources dedicated to Creatio (optional).
  - i. Go to the `clio` directory in Windows PowerShell.
  - ii. Run the `code .` command to open the structure of `clio` directory in Visual Studio Code.
  - iii. Redefine CPU resources.

For PostgreSQL

- a. Go to the `postgres` directory → `postgres-stateful-set.yaml` file.
- b. Redefine requested amount of `postgres-data` storage. To do this, go to the `spec` object → `volumeClaimTemplates` array of objects → object whose name string is `"postgres-data"` → `spec` object → `resources` object → `requests` map → `storage` string → change to `"10Gi."`
- c. Redefine requested amount of `postgres-backup-images` storage. To do this, go to the `spec` object → `volumeClaimTemplates` array of objects → object whose name string is `"postgres-backup-images"` → `spec` object → `resources` object → `requests` map → `storage` string → change to `"3Gi."`
- d. Open the `postgres-volumes.yaml` file.
- e. Redefine quantity of `postgres-data-pv` resource. To do this, go to the object whose name string is `"postgres-data-pv"` → `spec` object → `capacity` map → `storage` string → change to `"10Gi."`
- f. Redefine quantity of `postgres-backup-images-pv` resource. To do this, go to the object whose name string is `"postgres-backup-images-pv"` → `spec` object → `capacity` map → `storage` string → change to `"3Gi."`

For MSSQL

- a. Go to the `mssql` directory → `mssql-stateful-set.yaml` file.
- b. Redefine requested amount of `mssql-data` storage. To do this, go to the `spec` object → `volumeClaimTemplates` array of objects →

object whose name string is "mssql-data" → spec object  
→ resources object → requests map → storage string → change  
to "10Gi."

c. Open the `mssql-volumes.yaml` file.

d. Redefine quantity of `clio-mssql-data-pv` resource. To do this, go  
to the object whose name string is "clio-mssql-data-pv"  
→ spec object → capacity map → storage string → change to  
"10Gi."

## 2. Apply the changes.

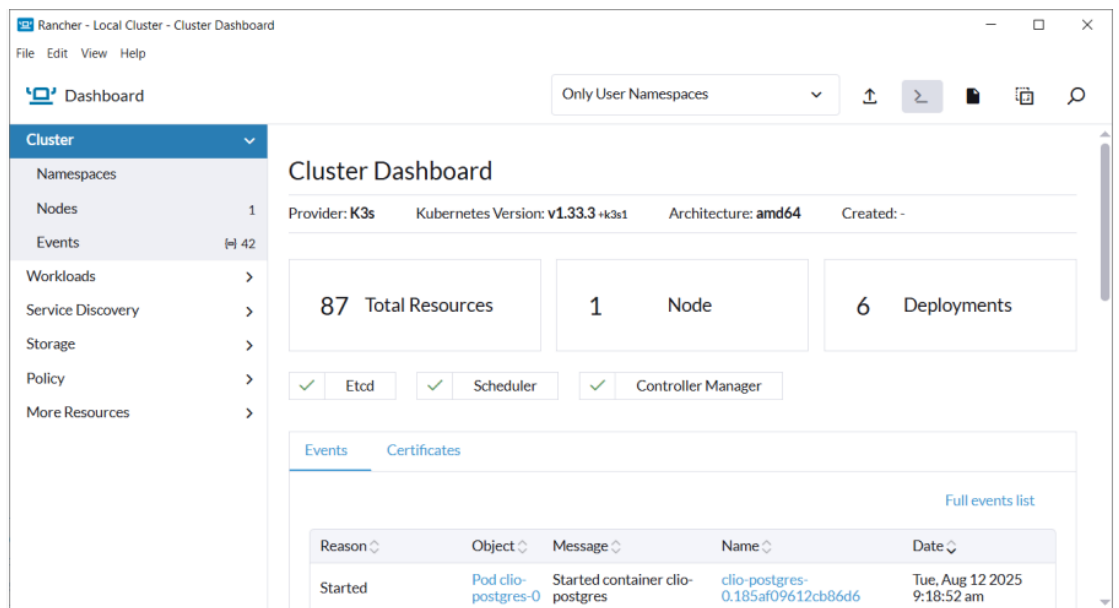
- i. Go to the Windows PowerShell that is run as administrator.
- ii. Go to the `infrastructure` directory.
- iii. Run the following commands.

```
# Create or update Kubernetes resources defined in the "clio-  
namespace.yaml" file.  
kubectl apply -f clio-namespace.yaml  
  
# Create or update a Kubernetes "StorageClass" defined in the "clio-  
storage-class.yaml" file.  
kubectl apply -f clio-storage-class.yaml  
  
# Apply Kubernetes resource files located in the "redis" directory.  
kubectl apply -f .\redis  
  
# Create or update Kubernetes resources located in the "postgres"  
directory.  
kubectl apply -f .\postgres  
  
# Create or update Kubernetes resources located in the "pgadmin"  
directory.  
kubectl apply -f .\pgadmin
```

## 3. Make sure Rancher Desktop includes required resources.

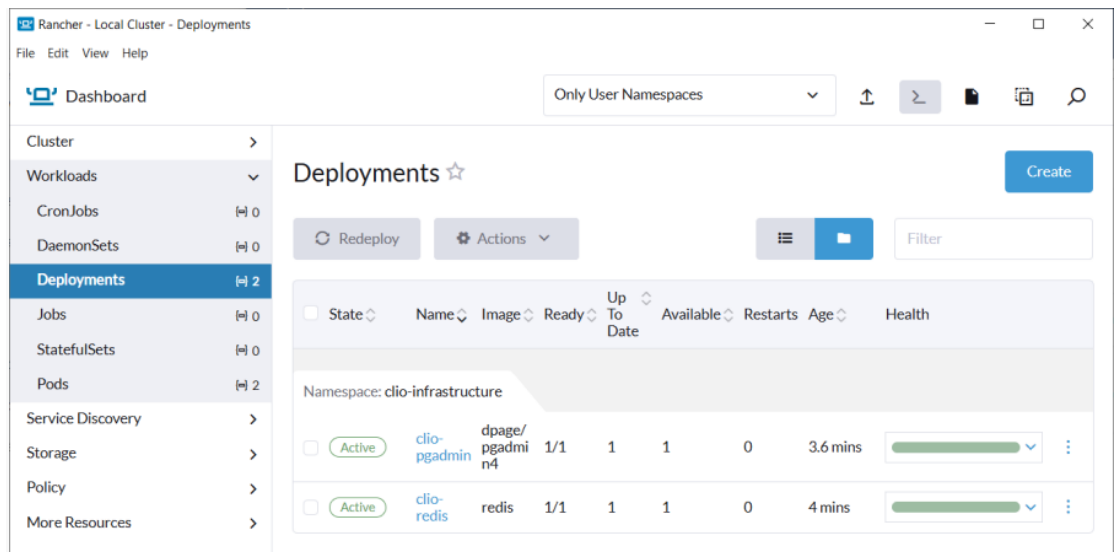
- i. Open the Rancher Desktop.
- ii. Click **Cluster Dashboard** to open the **Rancher - Local Cluster - Cluster Dashboard** window (Fig. 2).

Fig. 2 Cluster Dashboard window



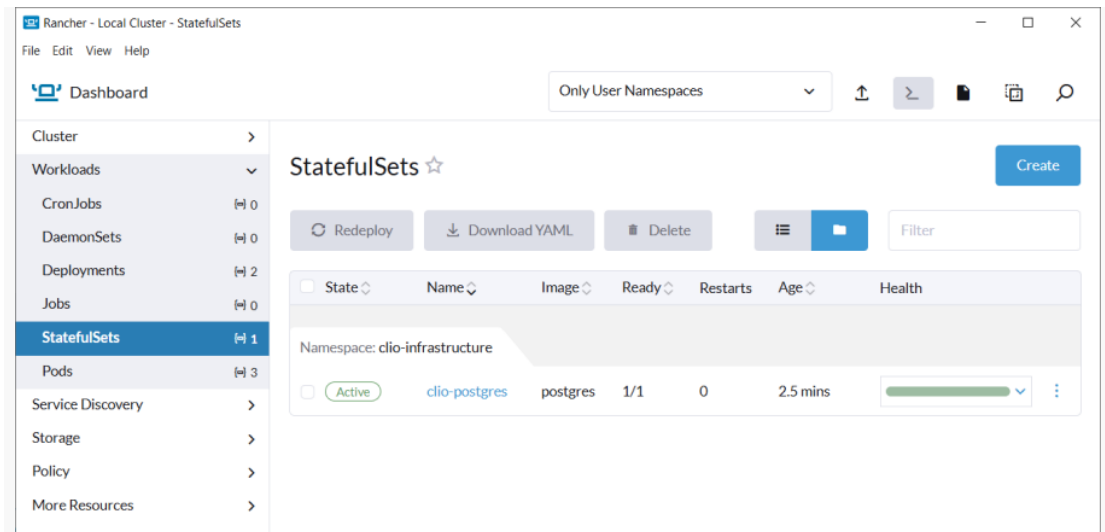
- iii. Click **Workloads** → **Deployment** to open the **Rancher - Local Cluster - Deployments** tab (Fig. 3).
- iv. Make sure the `clio-pgadmin` and `clio-redis` have the "Active" states.

Fig. 3 Deployments tab



- v. Click **StatefulSets** to open the **Rancher - Local Cluster - StatefulSets** tab (Fig. 4).
- vi. Make sure the `clio-postgres` has the "Active" state.

Fig. 4 StatefulSets tab



Otherwise, we recommend deleting all the files from the `postgres` directory:

- Go to the Windows PowerShell that is run as administrator.
- Go to the `infrastructure` directory.
- Run the `kubect1 delete -f postgres` command.
- Open the Rancher Desktop.
- Click **Cluster Dashboard** to open the **Rancher - Local Cluster - Cluster Dashboard** window (Fig. 2).
- Click **Storage** → **PersistentVolumes** to open the **Rancher - Local Cluster - PersistentVolumes** tab.
- Click **⋮** → **Delete**.
- Click **Storage** → **PersistentVolumeClaims** to open the **Rancher - Local Cluster - PersistentVolumeClaims** tab.
- Select the **postgres-data-clio-postgres-0** and **postgres-backup-images-clio-postgres-0** checkboxes → click **Delete** → confirm the action.
- Run the `kubect1 apply -f .\postgres` command to re-create Kubernetes resources located in the `postgres` directory.
- Make sure Rancher Desktop includes required resources.

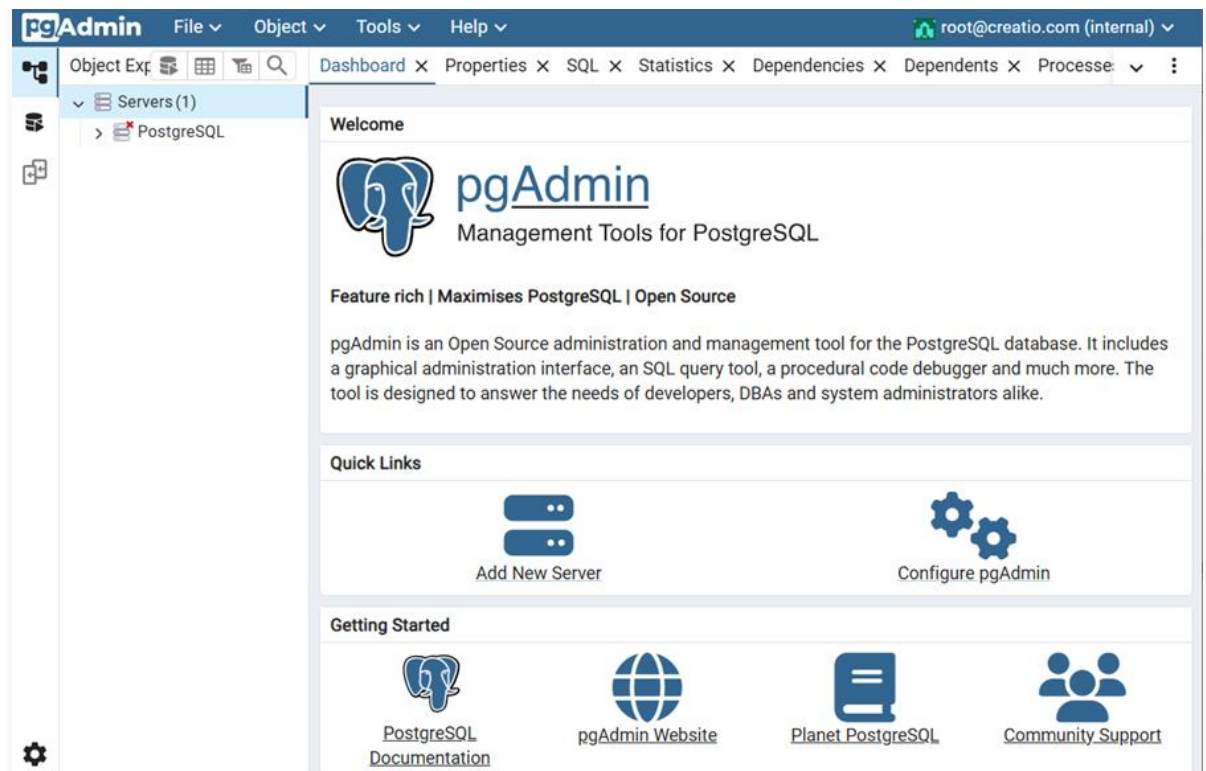
### 3. Verify that the pgAdmin GUI is accessible

1. **Find the port number** through which pgAdmin is accessible.
  - i. Open the structure of `clio` directory in Visual Studio Code.
  - ii. Go to the `pgadmin` directory → `pgadmin-services.yaml` file.
  - iii. Go to the `spec` object → `ports` array of objects → `port` integer value. The port number is "1080."
2. **Find the user credentials** to login.
  - i. Open the `pgadmin-secrets.yaml` file.
  - ii. Go to the `stringData` object.

The `PGADMIN_DEFAULT_EMAIL` string includes email to log in to pgAdmin website and is equal to "root@creatio.com."

The `PGADMIN_DEFAULT_PASSWORD` string includes password to log in to pgAdmin website and is equal to "root."
3. **Log in to** `http://localhost:1080` **URL** using user credentials. This opens the pgAdmin website (Fig. 5).

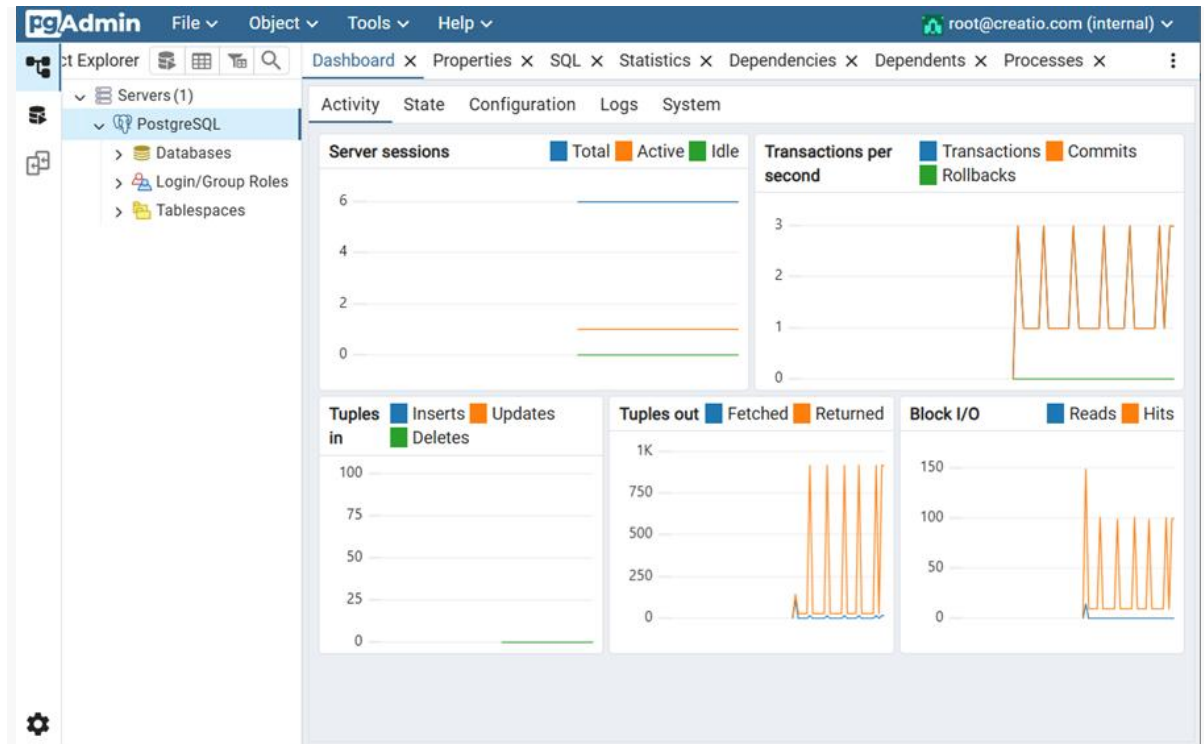
Fig. 5 pgAdmin website






4. Click **Servers** → **PostgreSQL** to open the **Connect to Server** window.
5. **Connect to the PostgreSQL server** using the same user password as the pgAdmin website. This opens the **Activity** tab for the connected server (Fig. 6).

Fig. 6 Activity tab



6. **Allow pgAdmin to display template databases.**
  - i. Click **File** → **Preferences** to open the **Preferences** tab.
  - ii. Go to the **Browser** → **Display**.
  - iii. Select the **Show template databases?** checkbox.
  - iv. Click . This opens the **Object explorer refresh required** window.
  - v. Click **Refresh**.

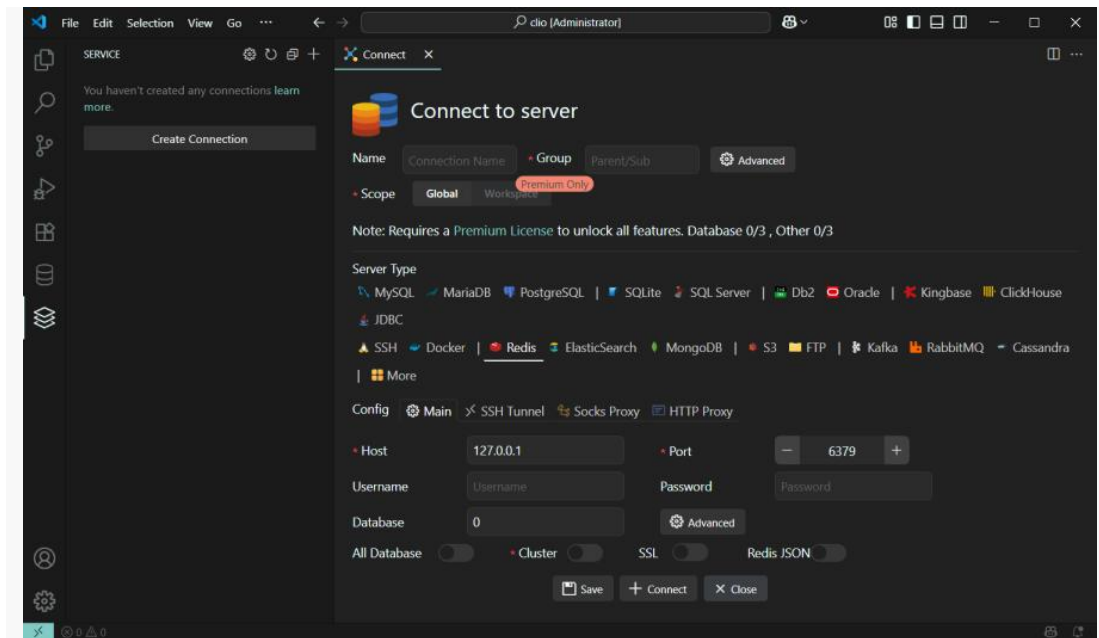
## 4. Set up Redis server

1. **Install Redis extension** (Redis Client for Visual Studio Code).  
Instructions: [official vendor website](#).

## 2. Connect to the server.

- i. Click **Create Connection** to open the **Connect to server** tab (Fig. 7).

Fig. 7 Connect to server tab




- ii. Fill out the connection parameters.

Parameter	Description
Name	An arbitrary name for Redis server. For example, "Redis-Clio."

- iii. Leave default values for other connection parameters.

- iv. Click  Save.

Out of the box, Redis server supports 99 databases. To **check whether the database is supported by the Redis server**:

1. Click  to open the Redis terminal.
2. Run the `select SomeDatabaseIndex` command.

**As a result:**

- If the terminal returns "OK" response, a database whose number is `SomeDatabaseIndex` can be supported by the Redis server.
- If the terminal returns "ERR DB index is out of range" response, a database whose number is `SomeDatabaseIndex` cannot be supported by the Redis server.

## 5. Select the way to deploy Creatio

Clio lets you deploy Creatio in the following ways:

- using File Explorer context menu
- using terminal

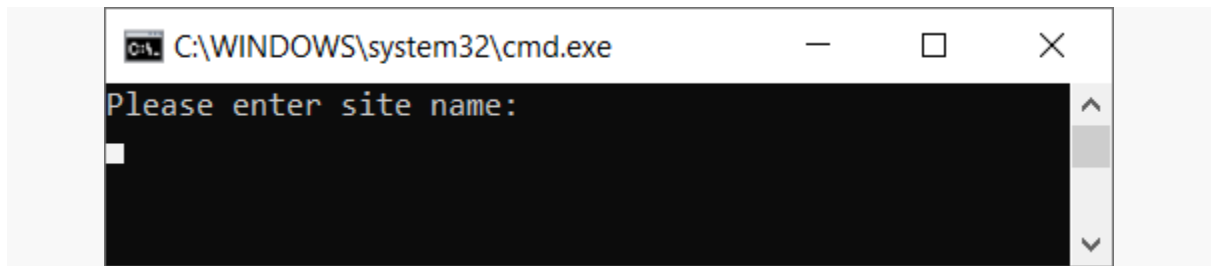
Before you select the way to deploy Creatio:

1. **Find the path and name of the IIS root directory** where Clio is deployed. The directory path and name are specified in the `iis-clio-root-path` property of the `C:\Users\SomeWindowsUser\AppData\Local\creatio\clio\appsettings.json` file. Out of the box, `C:\inetpub\wwwroot\clio`.
2. **Create the IIS root directory.**
3. **Make sure that IIS\_IUSRS user has the Full control checkbox selected** for the IIS root directory.
4. **Contact Creatio support** to receive Creatio setup zip archive. We recommend using PostgreSQL database.
5. **Save Creatio setup zip archive** to the directory on your local device. The directory path is specified in the `creatio-products` property of the `C:\Users\SomeWindowsUser\AppData\Local\creatio\clio\appsettings.json` file. Out of the box, `C:\CreatioProductBuild`.

### Deploy Creatio using File Explorer context menu

1. **Open the directory that has Creatio setup zip archive saved.**
2. **Right-click a Creatio setup zip archive → clio: deploy Creatio.** This opens the Windows command prompt (Fig. 8).

Fig. 8 Deploy Creatio using Clio



If context menu does not include Clio options, re-register Clio. To do this:

- i. Go to the Windows PowerShell that is run as administrator.
- ii. Run the `clio register` command.

3. **Fill out the parameters** to deploy Creatio.

Parameter	Description
Please enter site name	An arbitrary name of the Creatio to deploy. For example, "DevEnv."
Please enter site port, Max value - 65535: (recommended range between 40000 and 40100)	Port to access deployed Creatio website. For example, "40015."

4. **Click Enter** to run Creatio deploying. Operation might take some time.

## Deploy Creatio using terminal

1. **Open the terminal of Visual Studio Code** or **run the Windows PowerShell** as administrator.
2. **Run the** `clio deploy-creatio --SiteName SomeCreatioWebsiteName --SitePort SomePort --ZipFile PathToSomeCreatioArchive` **command**, where:
  - `SomeCreatioWebsiteName` is an arbitrary name of your Creatio website.
  - `SomePort` is the port number to deploy Creatio instance.
  - `PathToSomeCreatioArchive` is the full path to the Creatio setup zip archive including archive name.

This runs Creatio deploying. Operation might take some time.

**As a result:**

- Creatio instance will be deployed.
- Creatio login page will be opened in the browser. Log in to Creatio using your user credentials.

Now you can start using Clio and execute different operation with Creatio using Clio directly from Visual Studio Code based on your business goals. Instructions: [Use Clio](#).

---

## See also

[Official Clio documentation \(GitHub\)](#)

[Clio explorer extension for Visual Studio Code](#)

[Official Redis website](#)

[Install Clio](#)

[Use Clio](#)

[Clio tutorials \(YouTube\)](#)