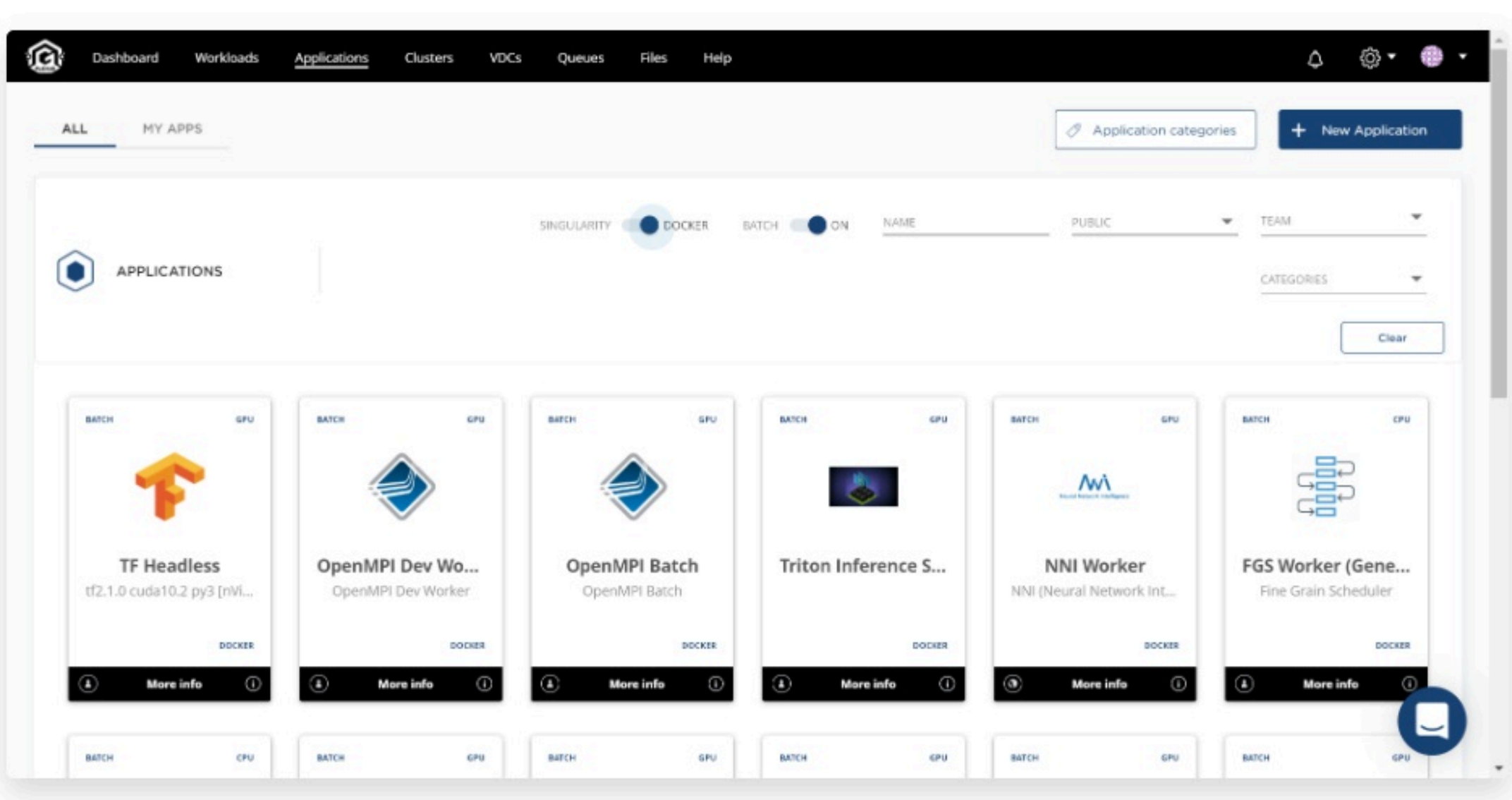
Step 3 - Create an application using Singularity

Contents

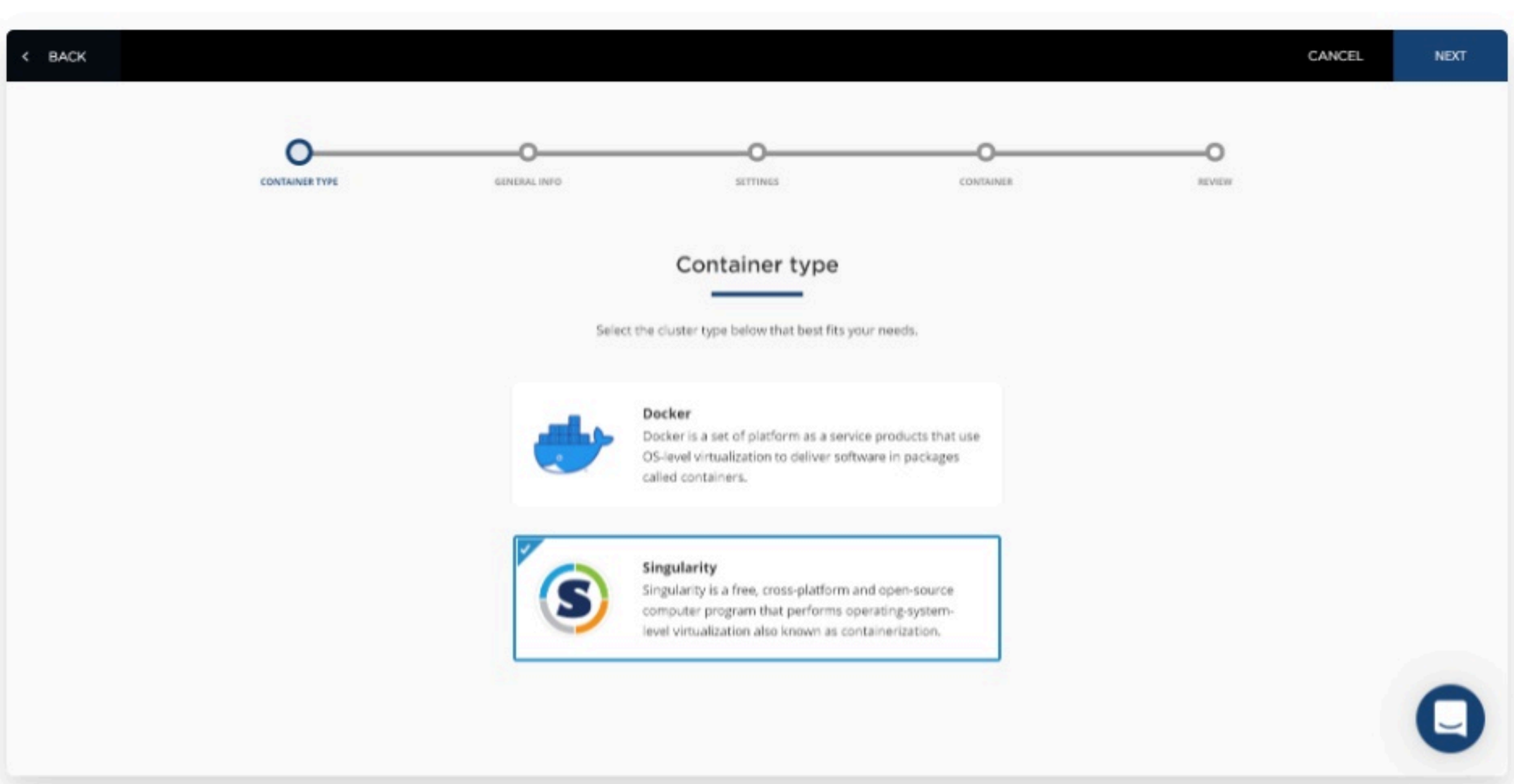
Step 3 - Create an application using Singularity

This article describes how to create a new Singularity application for Plexus.

1. In **Applications**, click **+ New Application** (The **New Application** function is available to **Administrator** and **Developer** users of Plexus)



2. Select the **Singularity** container type, and then click **NEXT**

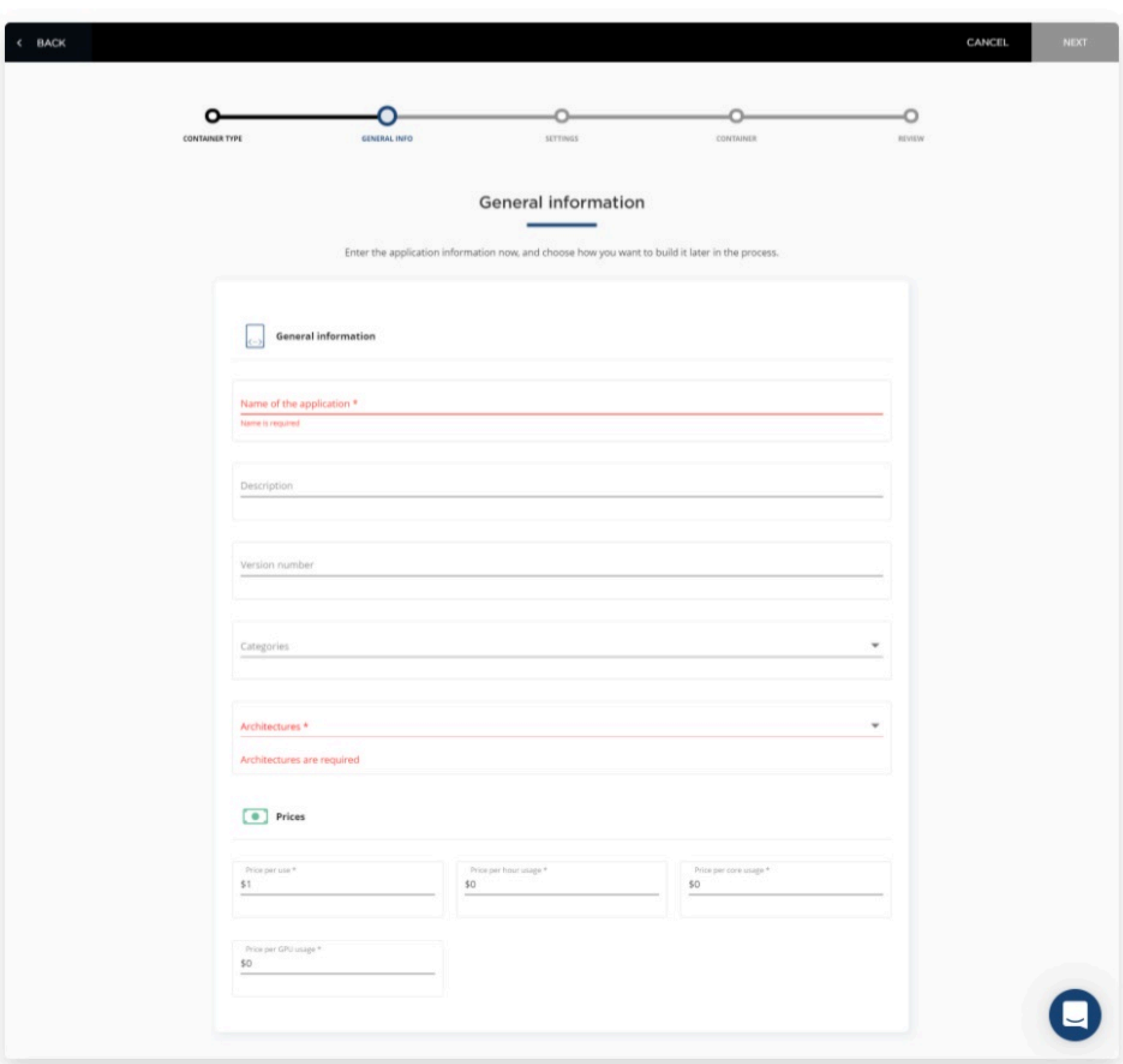


3. Type a **Name of the application** (required), and optionally, a **Description**, **Version number** and **Categories**.

NOTE: The name must be lowercase and contain only alphanumeric characters, or 'dash'. Spaces are not permitted).

In **Architectures** (required), select either **X86_64** or **aarch64.**

NOTE: AArch64 runtime performance is comparable to X64 for integer-based workloads, but AArch64 uses less energy.



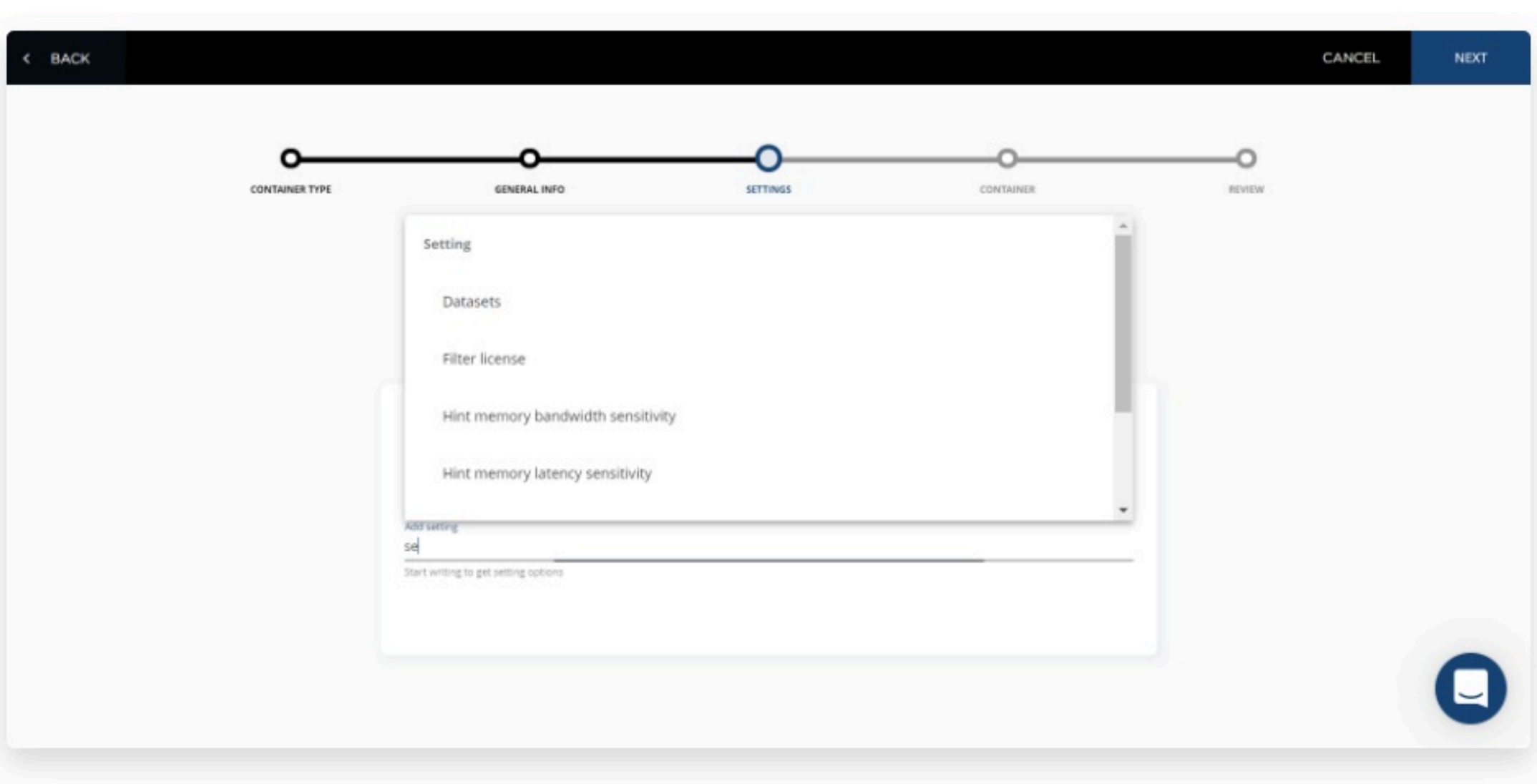
You can also provide pricing information for your app based on:

- Price per use
- Price per hour of usage
- Price per core usage.

The default pricing is $1 per use.

Click **NEXT**

4. On the **Settings** page you can *optionally* add metadata fields to the application, which can subsequently be used by scripts written against the API for a variety of filtering and metadata related activity. By clicking on the drop-down menu you will be able to see the range of metadata fields you can add – and there is no limit to the number of fields you can create.

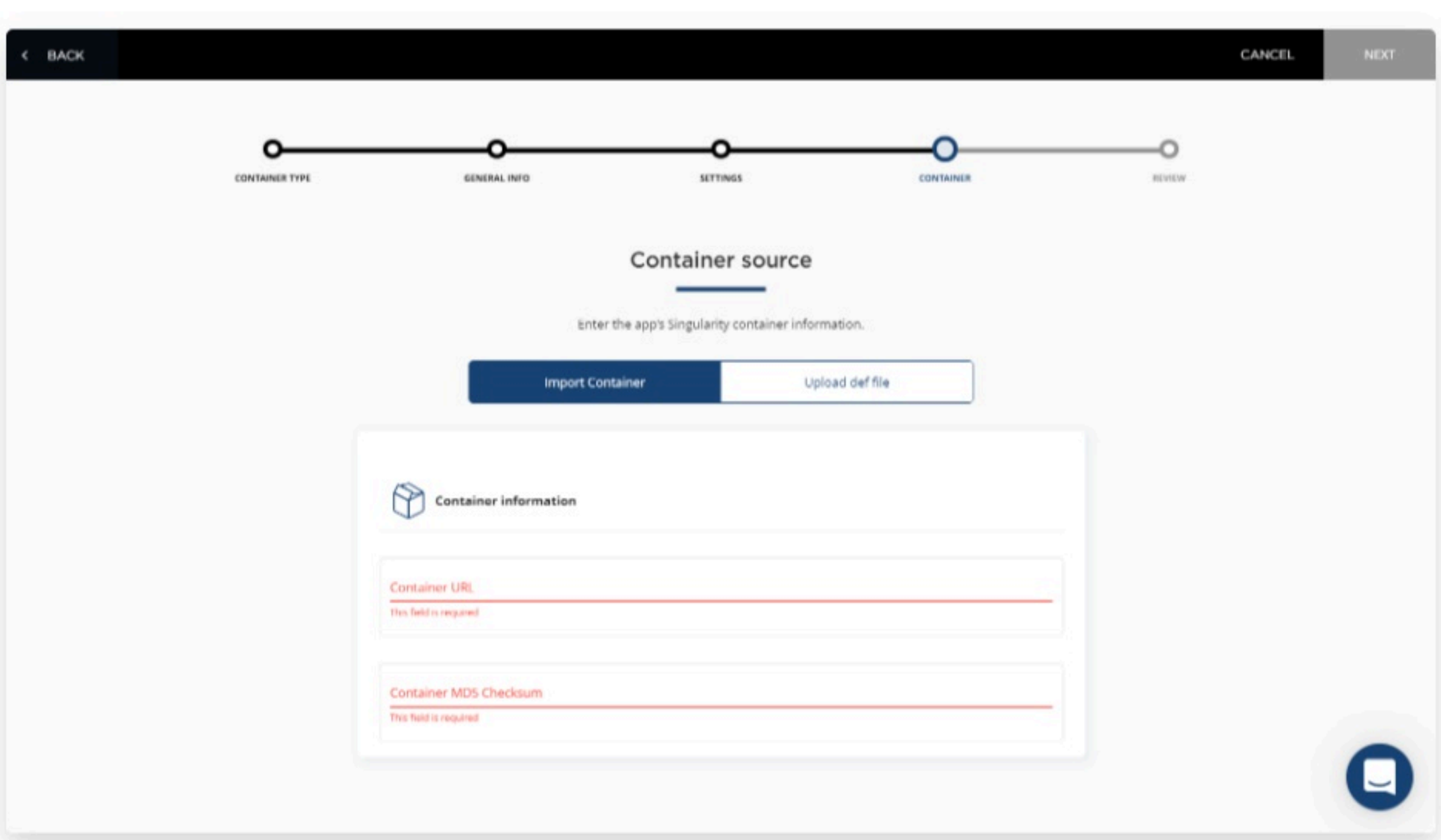


5. Click **NEXT** and you will see the **Create Container** page, which will show you the two different methods for creating the container for your app.

Two methods for building the new application

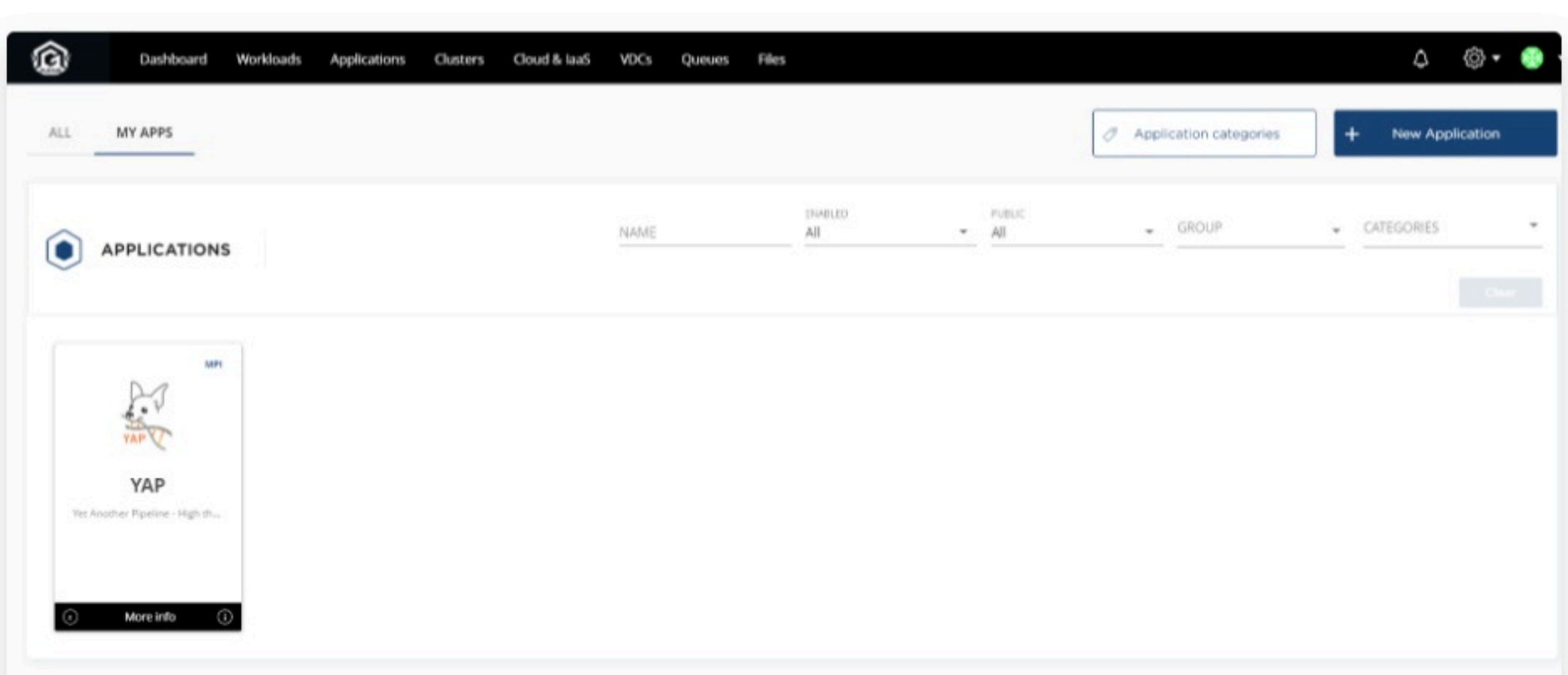
Method 1: Import Container

If you already have a Singularity or Docker container available for your specific application, the easiest method to add your app to Plexus marketplace is simply to import the container. The **Import Container** option lets you specify an existing container image file, which could be a Singularity image *or a Docker image*:

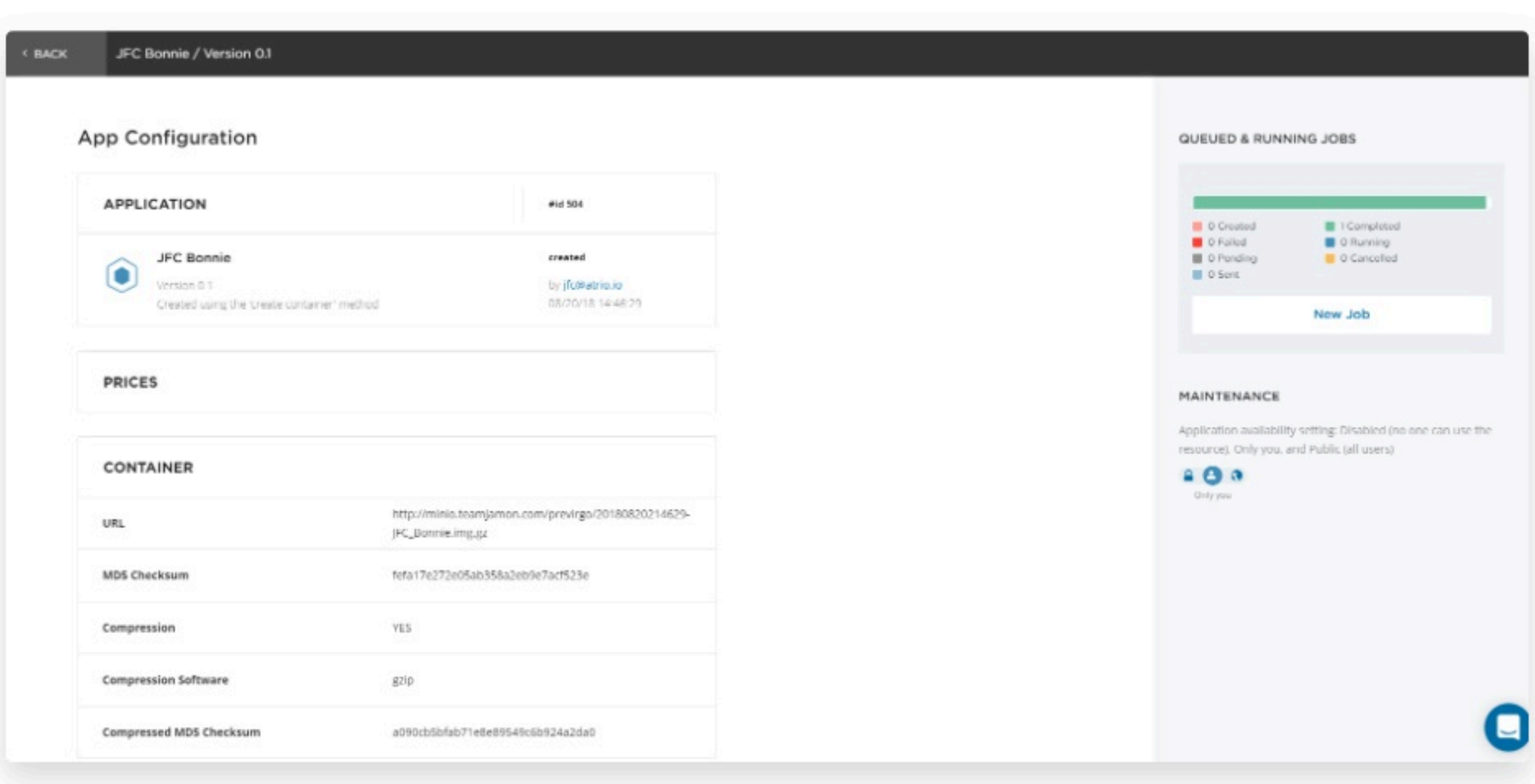


You will need to provide a link to the location where your Singularity or Docker container is stored, and, for Singularity - the specific container checksum. Make sure the URL you provide is fully accessible to the system, as there is currently no way to provide authentication credentials to retrieve the image

After you click on the Create button, your new imported container will be displayed on your '**My Apps** page:



and clicking on your new app panel will show you its configuration details (and eventually, its usage history in Plexus):

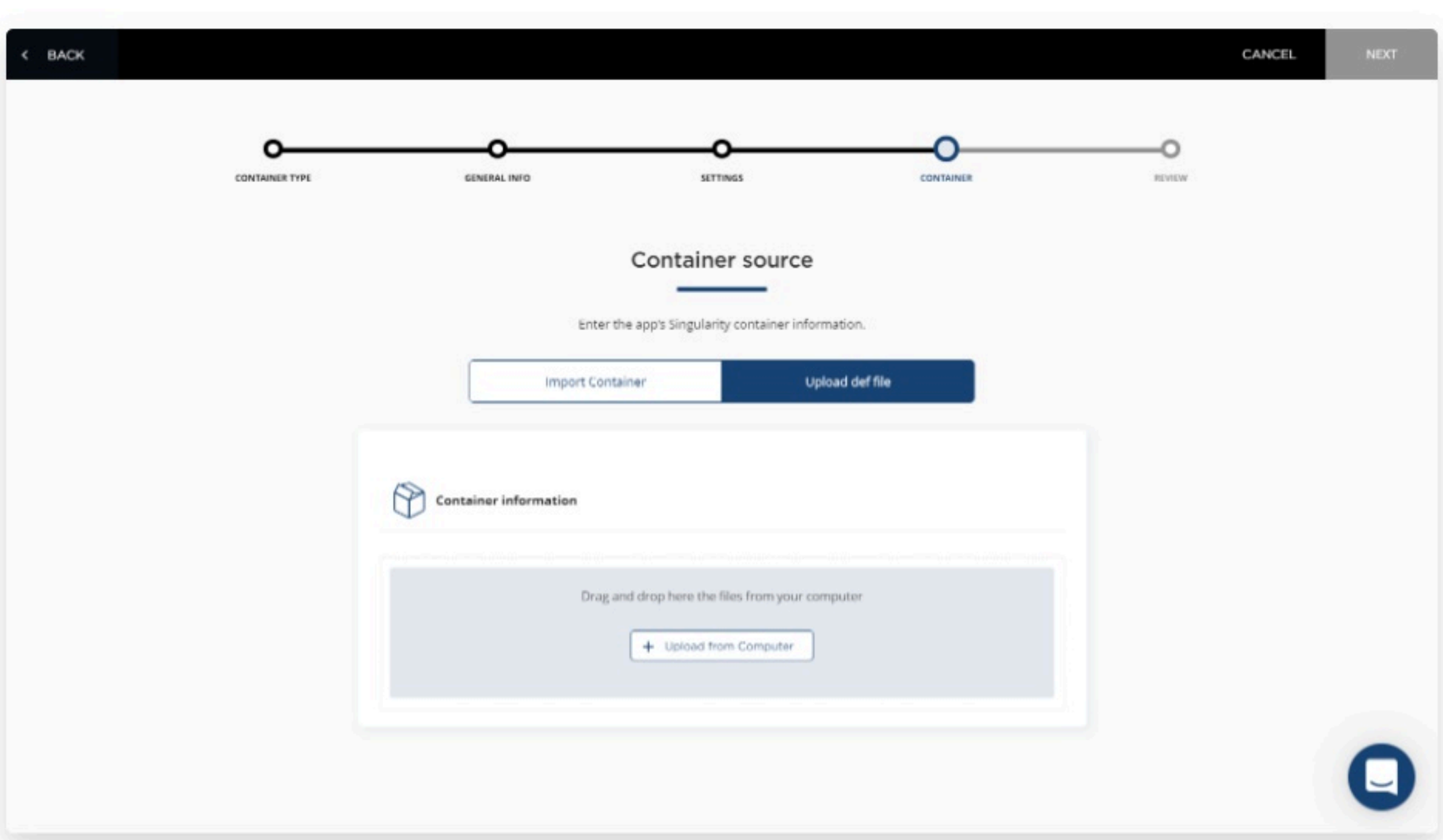


Notice that you can edit your app configuration, if required.

Also notice the **Maintenance** section on the right-hand side of the page – when you first create your new application it will be in the **disabled** state, and will not be usable; as the owner of this app, you can change its usage status to **Only You** or **Public** – which will make the app available to all users of this Plexus instance.

Method 2: Upload Def File

The **Upload Def File** option will let you submit a Singularity definition file, which will cause your specified container to be built by the Plexus build service, and then added to the application list on your **My Apps** page:



Clicking the **Next** button will start the build process, and you'll see your new app appear on your **My Apps** page. Build times can vary, depending on how busy the Plexus build service is – you should typically allow up to ten minutes or so, for this process to complete.

Example of a very simple def file:

```
BootStrap: debootstrap
OSVersion: xenial
MirrorURL: http://archive.ubuntu.com/ubuntu/

%post
  mkdir -p /app /liboverride /libfallback /prj /localscratch
/localflash /opt /scratch /LICENSE
  sed -i 's/main/main restricted universe/g' /etc/apt/sources.list
  apt-get update
  apt-get install -y wget vim git cmake gcc gfortran g++ curl
python autoconf bzip2 libtool libtool-bin bonnie++
  apt-get install -y libopenblas-dev libcurl4-openssl-dev
libopenmpi-dev openmpi-bin openmpi-common openmpi-doc openssh-
client openssh-server libssh-dev wget vim
  apt-get clean
  locale-gen en_US.UTF-8
  wget http://illposed.net/stream2.tar.gz
  tar zxvf stream2.tar.gz && rm -f stream2.tar.gz
  make -C stream2 && make -C stream2 install
  rm -rf /usr/share
  rm -rf /usr/src
  cat << EOF > /usr/bin/bonnie++.bench
#!/bin/bash
p=\`cat /proc/cpuinfo | grep 'physical id\|core id' | paste -d'
' - - | sort -u | wc -l\`
j=0
while test \$j -lt \$p; do
  bonnie++ -r 1000 -b &
  j=\$((\$j + 1))
done
```

```
wait
EOF
  cat << EOF > /usr/bin/stream2.bench
#!/bin/bash
p=\`cat /proc/cpuinfo | grep 'physical id\|core id' | paste -d'
' - - | sort -u | wc -l\`
j=0
while test \$j -lt \$p; do
  stream2 &
  j=\$((\$j + 1))
done
wait
EOF
  chmod a+rx /usr/bin/stream2.bench
  chmod a+rx /usr/bin/bonnie++.bench


%runscript
  TESTTIME=300   # approximate test total run time
  T0=`date +%s`
  DT=0
  while test $TESTTIME -gt 10; do
    timeout $TESTTIME  bonnie++.bench || echo "timed out"
    R=$?
    T1=`date +%s`
    DT=`expr $T1 - $T0`
    TESTTIME=`expr $TESTTIME - $DT`
  done
  exit $R
```

For more information about how to work with Singularity containers, see Singularity documentation at https://sylabs.io/guides/3.6/user-guide/.

About Core Scientific

Core Scientific is a leader in infrastructure and software solutions for Artificial Intelligence and Blockchain.