创建LNP应用（Linux+Nginx+PHP-7.3.3）

1. 制作LNP镜像

2. 登录URLOS添加LNP镜像

3. 添加LNP应用

4. 添加LNP应用模板

5. 添加LNP应用变量

6. 测试LNP应用

1. **制作LNP镜像**

   用ssh登录自己的宿主机器，下载并运行官方镜像。

   ```
   docker run -itd php:7.3.3-fpm-stretch bash
   ```

   进入容器。更新镜像源。

   ```
   set -ex \ && sed -i 's@security.debian.org@mirrors.aliyun.com@' /etc/apt/sources.list

   set -ex \ && sed -i 's@deb.debian.org@mirrors.aliyun.com@' /etc/apt/sources.list

   apt-get update
   ```

   安装nginx程序。（默认安装目录：/etc/nginx）

   ```
   apt-get install -y nginx
   ```

   官方镜像默认是没有ps -ef 命令，因此需要手动安装

   ```
   apt-get install -y procps
   ```

   安装PHP扩展。

   安装php自带的一些扩展时，可以使用docker-php-ext-configure和docker-php-ext-install。例如我们要安装pdo_mysql

   ```
   docker-php-ext-configure pdo_mysql
   docker-php-ext-install pdo_mysql
   ```

然后使用 php -m查看我们的扩展是否安装成功。使用这种方式安装，系统会自动生成一个配置文件，提供给php加载，使用命令查看：

```
ls -l /usr/local/etc/php/conf.d/
```

gd扩展安装

```
apt-get install -y libfreetype6-dev \
        libjpeg62-turbo-dev \
        libpng-dev
docker-php-ext-configure gd --with-freetype-dir=/usr/include/ --with-jpeg-
dir=/usr/include/
docker-php-ext-install gd
```

如果需要安装memcached、redis扩展，则需要下载扩展到容器，然后手动编译安装。地址：

https://pecl.php.net/package/memcached

https://pecl.php.net/package/redis

memcached扩展安装：

```
curl -O https://pecl.php.net/get/memcached-3.1.3.tgz
tar xf memcached-3.1.3.tgz && cd memcached-3.1.3.tgz
phpize
./configure
```

编译过程中若出现以下错误提示。

```
checking for libmemcached location... configure: error: memcached support requires
libmemcached. Use --with-libmemcached-dir=<DIR> to specify the prefix where
libmemcached headers and library are located
```

则执行安装命令，然后重新编译安装memcached扩展。

```
apt-get install -y libmemcached-dev
./configure
make && make install
```

添加extension=memcached.so语句到php.ini文件。安装完成后通过命令查看扩展存放的位置

```
ls /usr/local/lib/php/extensions/no-debug-non-zts-20170718/
```

php安装目录：/usr/local/php

php.ini的配置文件目录：/usr/local/etc/php/。在这个目录下有两个文件：php.ini-development和php.ini-production。

因此，我们需要将php.ini-production文件重命名为php.ini。以后手动编译安装php扩展后需要添加extension=xx.so到php.ini。

启动Nginx和php，检查是否正常运行。

nginx && php-fpm -D

打包镜像并提交到hub.docker.com。

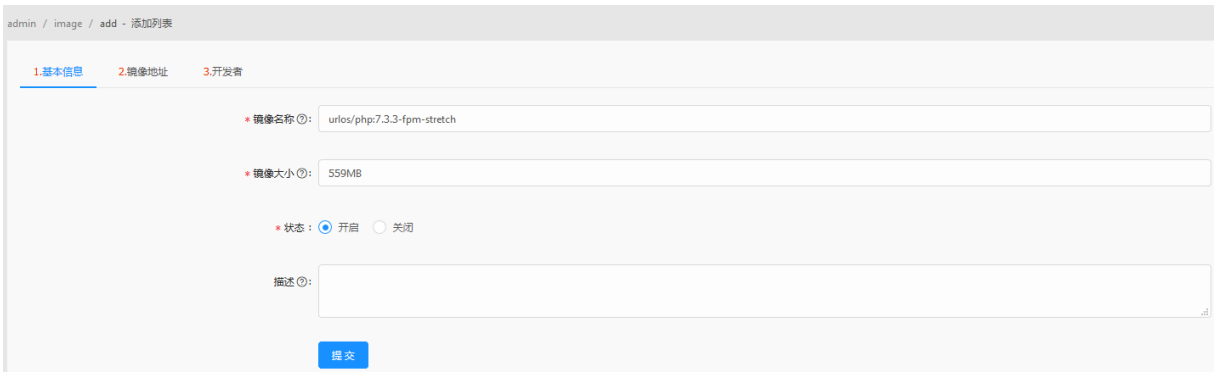
将当前的容器打包成镜像并上传到hub.docker.com。在打包成镜像之前，我们先将nginx、php-fpm关闭，删除一些不需要的应用以及清理一些安装的缓存文件，从而减小最终打包成镜像的大小。

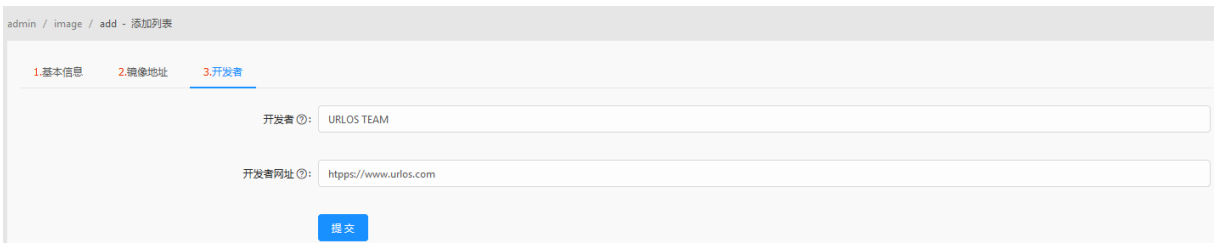
```
apt-get purge vim make
apt-get autoremove
apt-get autoclean
rm -f /usr/local/etc/php/conf.d/* #统一将php扩展写入到php.ini文件
```

2. **登录URLOS添加镜像**

注意：需要修改将/data/urlos/master-config/config.json文件的envType的值设置为dev

登录自己的URLOS，在左侧菜单栏选择镜像管理，然后点击右上角的添加按钮。

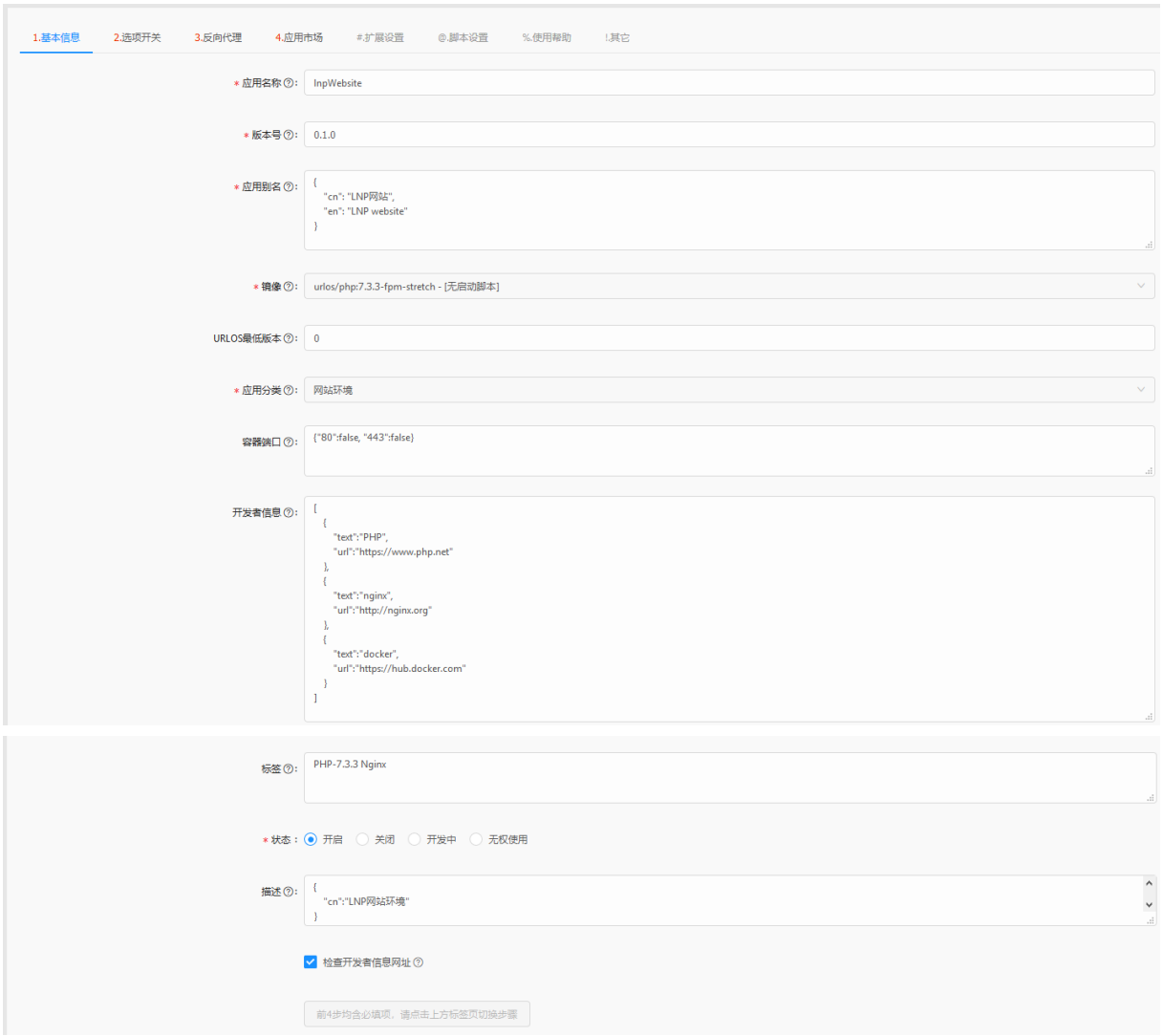




填写完成后，点击保存。

3. **创建LNP应用**

在左侧菜单中选择应用管理。然后点击右上角的添加应用按钮。



应用的基本信息如上图所示。

镜像：选择上一步骤添加的镜像。

URLOS最低版本号：如设置此选项则表示安装URLO的版本高于或者等于当前设置的值，才允许用户安装使用。

容器端口：容器启动时对外通信的端口，即参数-p。网站的80、443等端口默认是对外开发的，在这里可以不用设置。如必须特定端口时，设置的格式{"22":true}。

标签：应用标签多用于搜索场景。

选项开关注解：

固定节点运行：若勾选，则表示用户在安装此应用时（创建此应用的服务）需要选择安装在某个节点（云主机）。若取消勾选，则表示此应用安装在选择的集群(单容器运行也取消勾选)，可达到负载均衡，故障转移的效果。

单容器运行：若勾选，则表示安装此应用时，每个服务只运行一个容器。与固定节点运行配合使用，即固定节点运行时，则单容器运行。

允许特权允许：若勾选，则容器内的root拥有了真正的root权限（宿主机器的root）,在容器内部就可以做任何事情（包括修改宿主机器的文件，启动宿主机器其他容器，执行mount等操作），不建议勾选。

以root用户允许容器：这里的root用户是容器外部的一个普通用户，默认勾选。若容器内部的程序禁止root用户允许，则取消（如：MySQL）。

挂载存储目录：如需要从宿主机器挂载文件到容器，则勾选。即参数 -v。

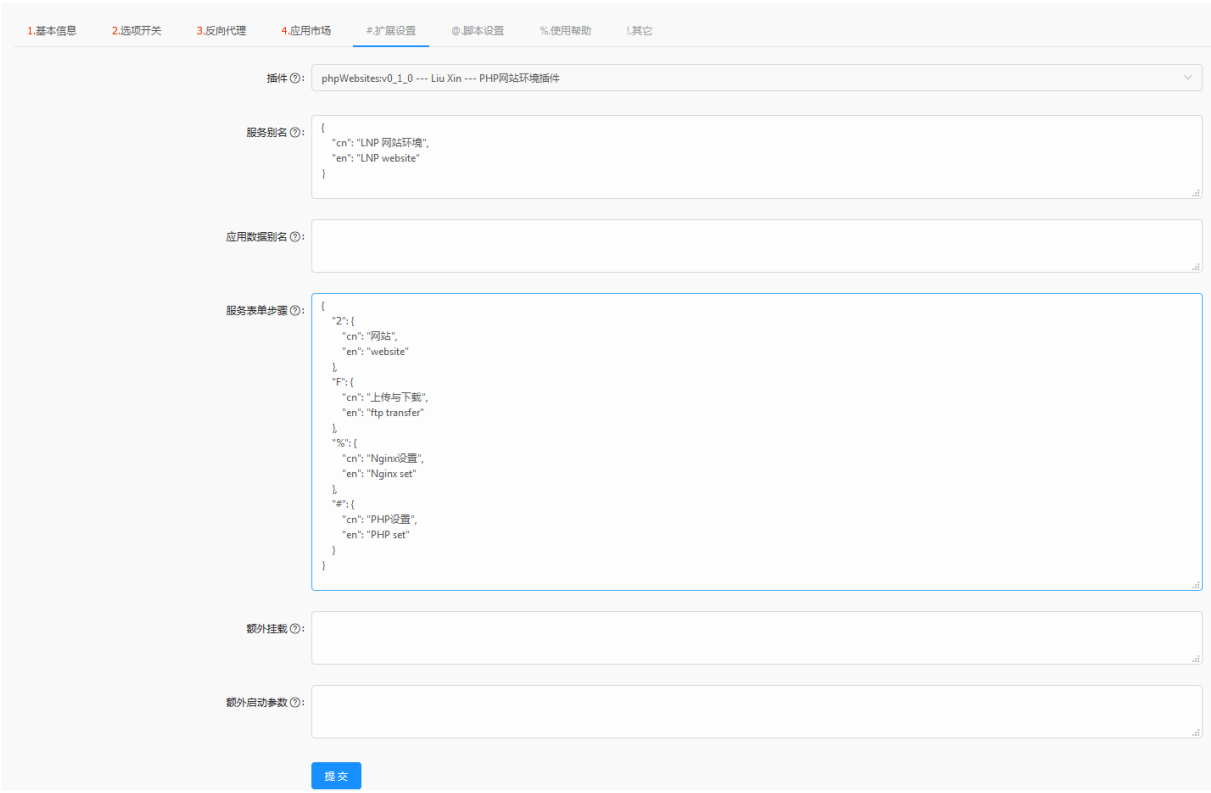
挂载时区定义文件：容器的时间与宿主机器的时间保持一致。

容器只读：禁止向容器写数据。

全局网络：允许同一集群不同容器网络的容器通信。

允许快照备份：勾选则允许执行快照备份（仅挂载了本地存储时有效）

开启反向代理，则可以实现多容器共享端口，反之则不能。



注解：

插件：由PHP语言编辑的脚本文件组成。插件的使用会让用户在安装应用（创建服务）时更便捷，更智能。这里选择phpWebSite:v0_1_0 --- Liu Xin ---php网站环境这个插件即可。（制作插件后续会有详细说明）

服务别名：创建服务时，在左上角显示的描述。

应用数据别名：创建服务完成后，服务产生的数据或者用户基于创建的服务需要添加新的数据，对这些数据管理取的名字，即为应用数据别名。（如：创建MySQL数据库服务，用户可以手动添加数据库，创建网站服务时也可以新增数据库。）

服务表单步骤：创建服务时，用户填写表单的步骤。（数字表示必填，其他符号表示选填）

额外挂载：将宿主机器的除存储目录外的其他目录挂载到容器。

额外启动参数：通过docker run运行容器时的额外参数，如：--add-host a.com:192.168.0.1

注解：

安装脚本：安装应用时需要执行的脚本命令。

```
test -d /etc/nginx/conf.d/ || mkdir -p /etc/nginx/conf.d/
```

启动脚本：需要启动程序的命令。

```
nginx
php-fpm -D
```

状态脚本：每隔2秒执行此脚本，用来检查程序是否正常允许。当前的脚本命令用来检查apache是否启动。

```
status1=0 && (ps -ef|grep "php-fpm"|grep "master process"|grep -v "grep") &&
status1=1;
status2=0 && (ps -ef|grep "nginx"|grep "master process"|grep -v "grep") &&
status2=1;
if [ ${status1} != 0 ] && [ ${status2} != 0 ]; then
    statusScriptResult=1
fi
```

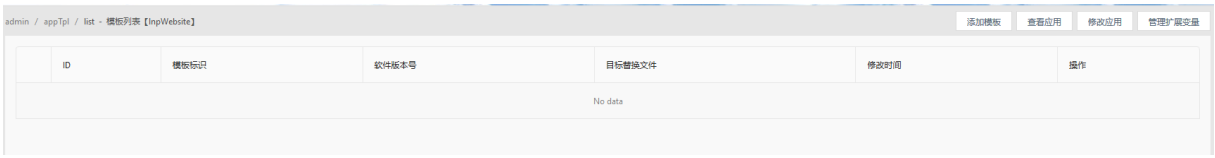
监控脚本：每隔1秒执行此脚本，检查状态脚本返回的结果判断程序是否正常允许。若异常，则执行退脚本。

```
{w:statusScript:w}
[ "$statusScriptResult" != 1 ]  && exit 1
```

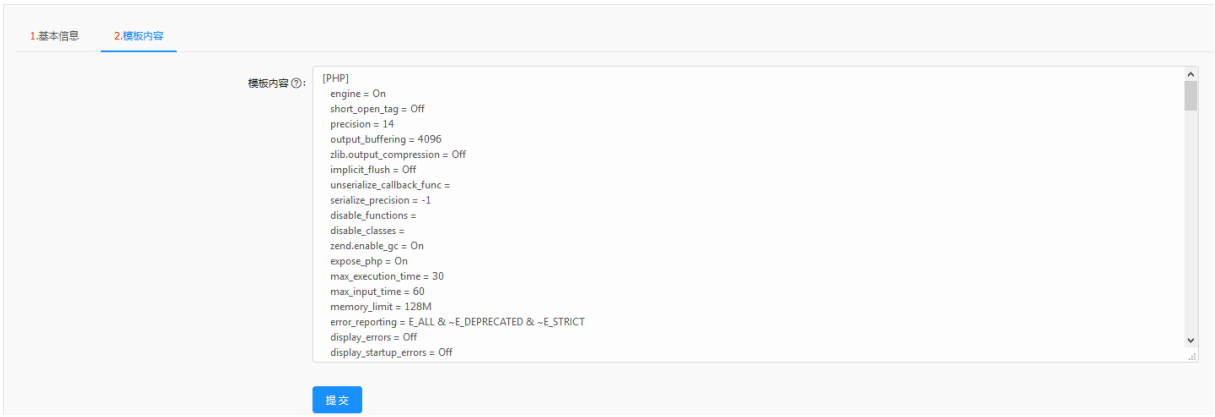
退出脚本：容器关闭时之前，执行的脚本。如同，我们关闭电脑时，系统会关闭正在允许的程序。

4. **添加LNP模板**

在这个应用，我们需要添加模板php.ini、vhost.conf，然后在这两个模板的参数设置一些变量，这样用户在安装应用时，就可以根据自己的需要动态的调整。（如：设置php的上传大小，最大内存等）那么如何添加模板呢？我们在应用管理列表中找到上述创建的应用，然后点击右侧的更多选择管理模板。



添加一个php.ini模板，然后在模板内容将php.ini文件内容复制进入，同时设置变量{w:upload_max_filesize:w}、{w:PHP_memory_limit:w}。



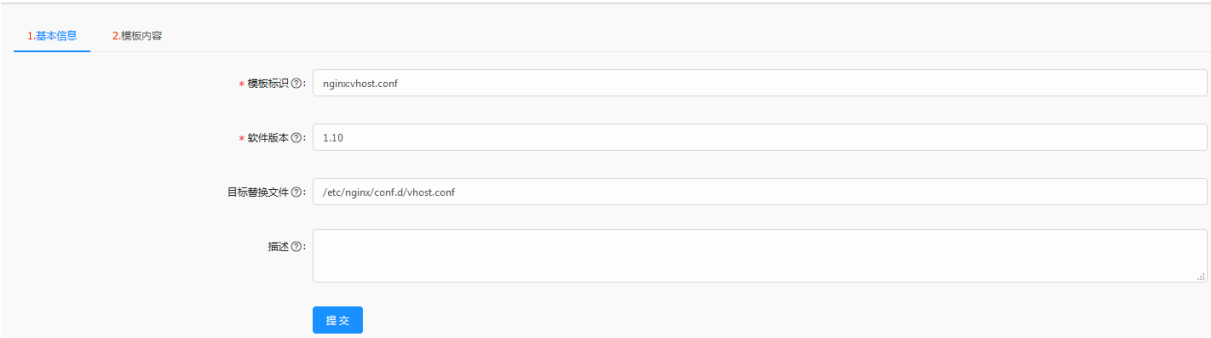

```
[PHP]
engine = On
short_open_tag = Off
precision = 14
output_buffering = 4096
zlib.output_compression = Off
implicit_flush = Off
unserialize_callback_func =
serialize_precision = -1
disable_functions =
disable_classes =
zend.enable_gc = On
expose_php = On
max_execution_time = 30
max_input_time = 60
memory_limit = 128M
error_reporting = E_ALL & ~E_DEPRECATED & ~E_STRICT
display_errors = Off
display_startup_errors = Off
log_errors = On
log_errors_max_len = 1024
ignore_repeated_errors = Off
ignore_repeated_source = Off
report_memleaks = On
html_errors = On
```

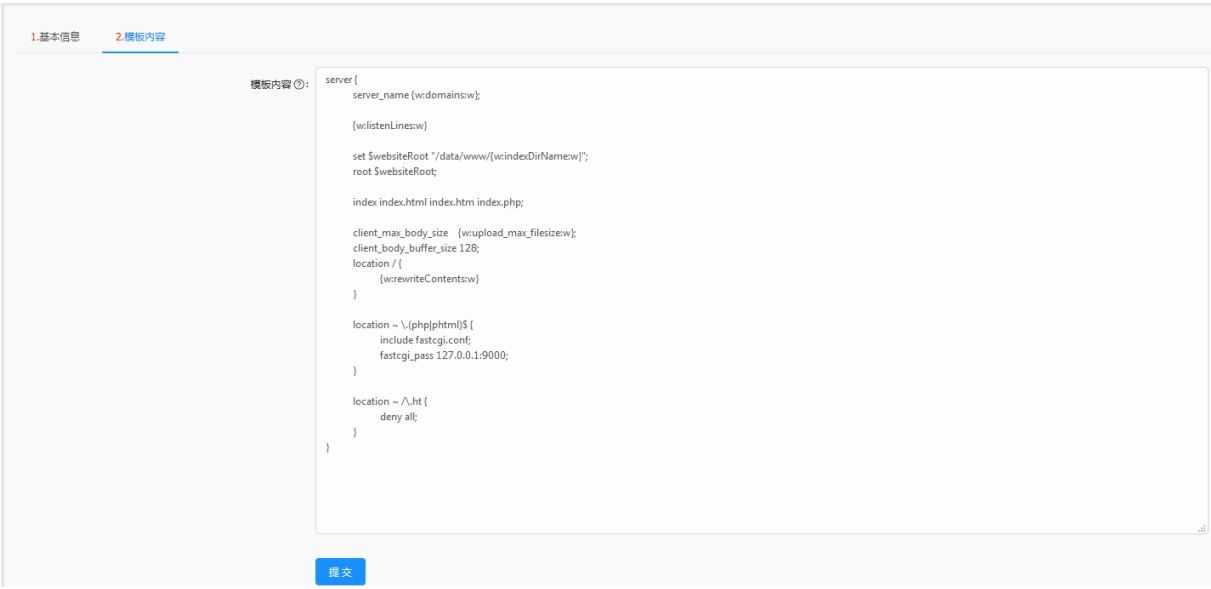
```
variables_order = "GPCS"
request_order = "GP"
register_argc_argv = Off
auto_globals_jit = On
post_max_size = {w:PHP_memory_limit:w}
auto_prepend_file =
auto_append_file =
default_mimetype = "text/html"
default_charset = "UTF-8"
doc_root =
user_dir =
enable_dl = Off
file_uploads = On
upload_max_filesize = {w:upload_max_filesize:w}
max_file_uploads = 20
allow_url_fopen = On
allow_url_include = Off
default_socket_timeout = 60
extension=gd.so
extension=memcached.so
extension=sockets.so
extension=mysqli.so
extension=pdo_mysql.so
[CLI Server]
cli_server.color = On
[Date]
[filter]
[iconv]
[imap]
[intl]
[sqlite3]
[Pcre]
[Pdo]
[Pdo_mysql]
pdo_mysql.default_socket=
[Phar]
[mail function]
SMTP = localhost
smtp_port = 25
mail.add_x_header = Off
[ODBC]
odbc.allow_persistent = On
odbc.check_persistent = On
odbc.max_persistent = -1
odbc.max_links = -1
odbc.defaultlrl = 4096
odbc.defaultbinmode = 1
[Interbase]
ibase.allow_persistent = 1
ibase.max_persistent = -1
ibase.max_links = -1
ibase.timestampformat = "%Y-%m-%d %H:%M:%S"
ibase.dateformat = "%Y-%m-%d"
```

```
ibase.timeformat = "%H:%M:%S"
[MySQLi]
mysqli.max_persistent = -1
mysqli.allow_persistent = On
mysqli.max_links = -1
mysqli.default_port = 3306
mysqli.default_socket =
mysqli.default_host =
mysqli.default_user =
mysqli.default_pw =
mysqli.reconnect = Off
[mysqlnd]
mysqlnd.collect_statistics = On
mysqlnd.collect_memory_statistics = Off
[OCI8]
[PostgreSQL]
pgsql.allow_persistent = On
pgsql.auto_reset_persistent = Off
pgsql.max_persistent = -1
pgsql.max_links = -1
pgsql.ignore_notice = 0
pgsql.log_notice = 0
[bcmath]
bcmath.scale = 0
[browscap]
[Session]
session.save_handler = files
session.use_strict_mode = 0
session.use_cookies = 1
session.use_only_cookies = 1
session.name = PHPSESSID
session.auto_start = 0
session.cookie_lifetime = 0
session.cookie_path = /
session.cookie_domain =
session.cookie_httponly =
session.cookie_samesite =
session.serialize_handler = php
session.gc_probability = 1
session.gc_divisor = 1000
session.gc_maxlifetime = 1440
session.referer_check =
session.cache_limiter = nocache
session.cache_expire = 180
session.use_trans_sid = 0
session.sid_length = 26
session.trans_sid_tags = "a=href,area=href,frame=src,form="
session.sid_bits_per_character = 5
[Assertion]
zend.assertions = -1
[COM]
[mbstring]
[gd]
```

```
[exif]
[Tidy]
tidy.clean_output = Off
[soap]
soap.wsdl_cache_enabled=1
soap.wsdl_cache_dir="/tmp"
soap.wsdl_cache_ttl=86400
soap.wsdl_cache_limit = 5
[sysvshm]
[ldap]
ldap.max_links = -1
[dba]
[opcache]
[curl]
[openssl]
```

添加Nginx的虚拟站点配置vhost.conf模板。






```
server {
    server_name {w:domains:w};
    {w:listenLines:w}

   set $websiteRoot "/data/www/{w:indexDirName:w}";
   root $websiteRoot;

   index index.html index.htm index.php;
```

```
    client_max_body_size    {w:upload_max_filesize:w};
    client_body_buffer_size 128;

    location / {
        {w:rewriteContents:w}
    }

    location ~ \.(php|phtml)$ {
        include fastcgi.conf;
        fastcgi_pass 127.0.0.1:9000;
    }

    location ~ /\.ht {
        deny all;
    }
}
```

5. **添加变量**

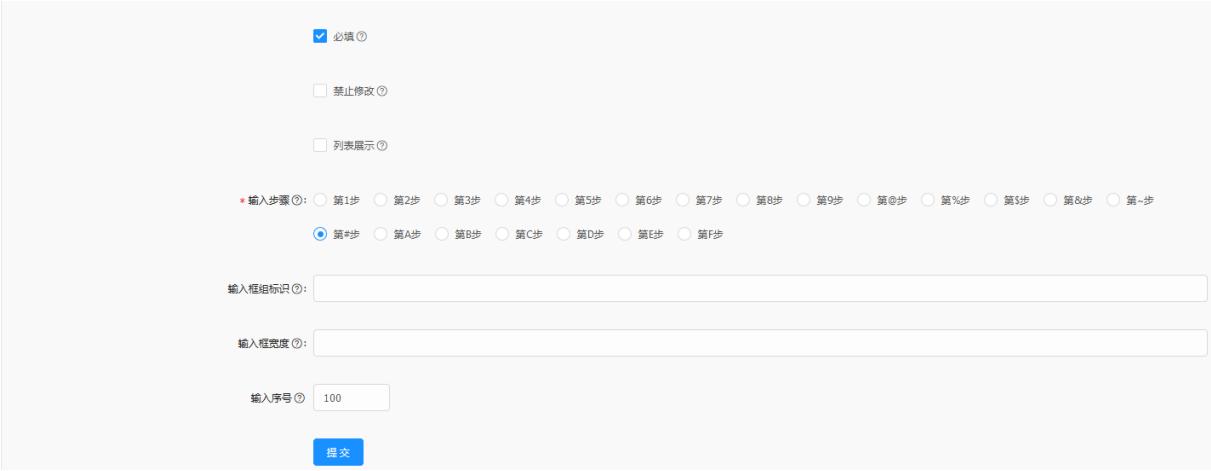
变量分为：环境变量、数据变量和扩展变量。在这里只需要添加扩展变量。

环境变量：在操作系统中用来指定操作系统运行环境的一些参数，与平常使用的环境变量相同。有时容器启动需要设置一些参数提供给容器内部的程序。如：MySQL容器启动时可以设置MYSQL_ROOT和MYSQL_ROOT_PASSWORD。

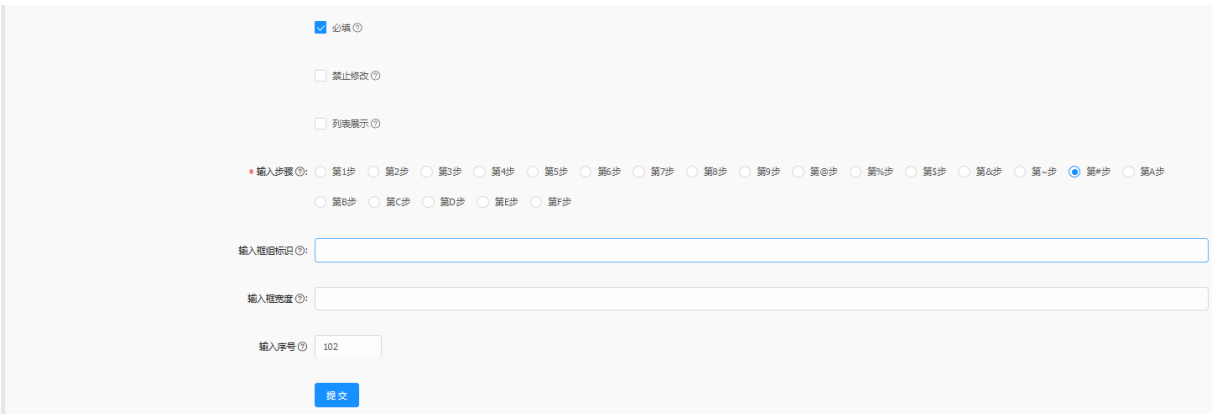
数据变量：添加存储数据时设置的一些参数。如：往MySQL数据服务添加数据库时，需要填写dbName，dbPassword，status，charset。具体可以使用可以创建MySQL服务，然后在管理数据库中添加数据库。

扩展变量：即普通变量。如：上述在模板中设置的变量{w:upload_max_filesize:w}、{w:PHP_memory_limit:w}。变量的格式：{w:变量名:w}。

添加PHP最大内存变量：PHP_memory_limit

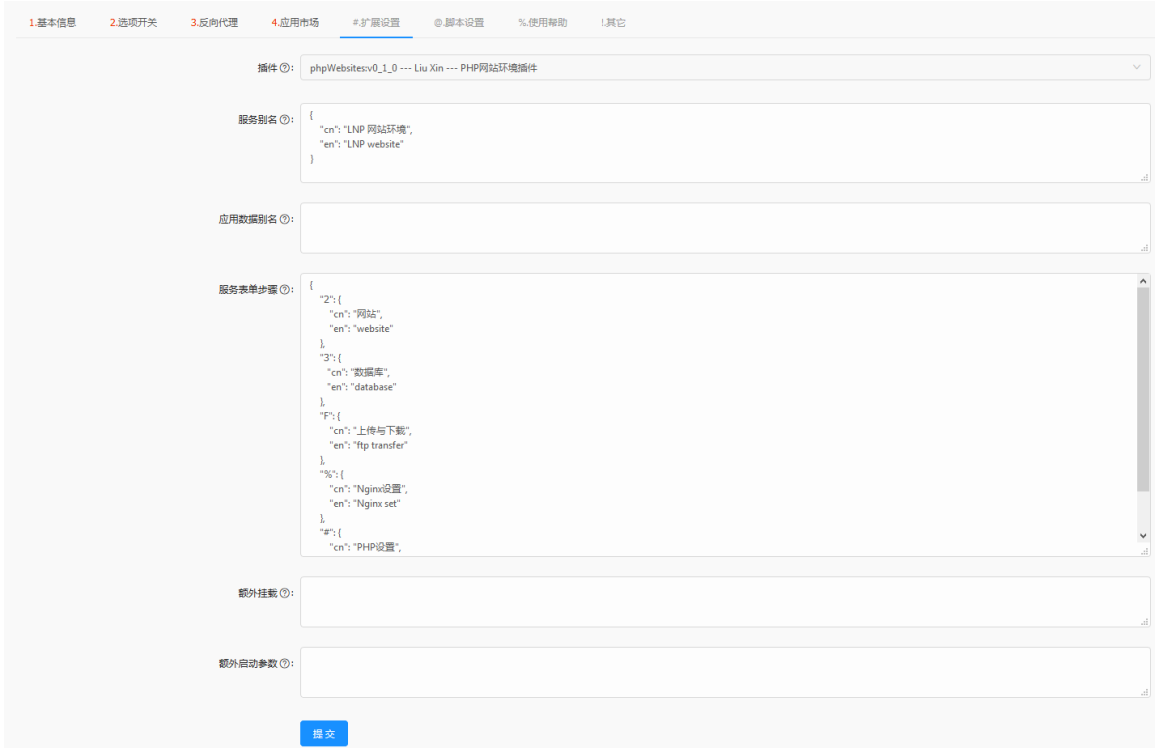


添加上传大小限制：upload_max_filesize

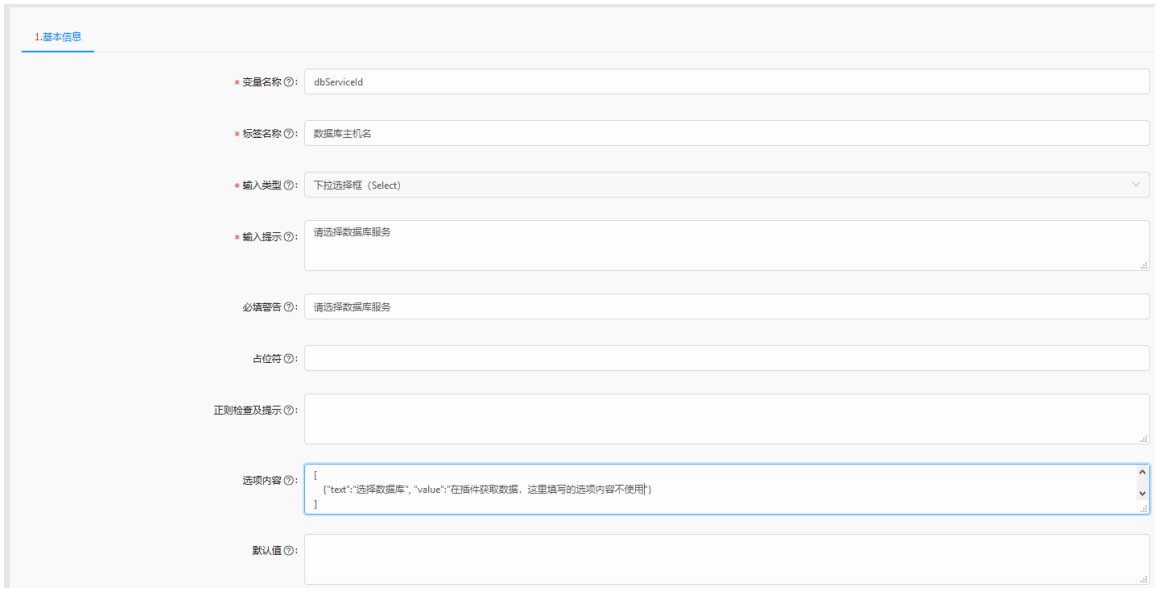


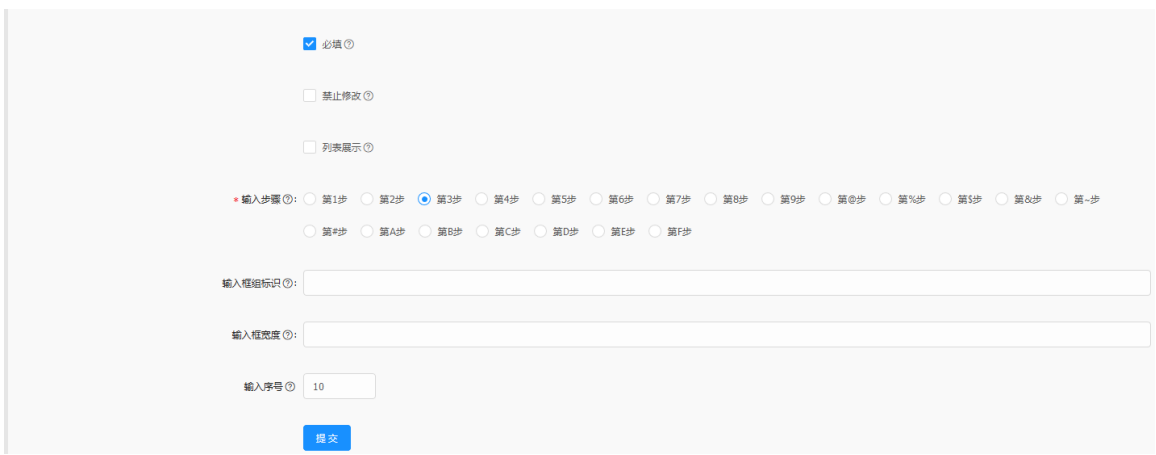
6. 创建LNP服务，部署完成后，打开浏览器访问页面。
7. 若要LNP应用支持MySQL数据库，请执行如下操作。

   7.1 修改LNP应用的扩展设置选项的服务表单步骤的内容增加数据库步骤。如图：

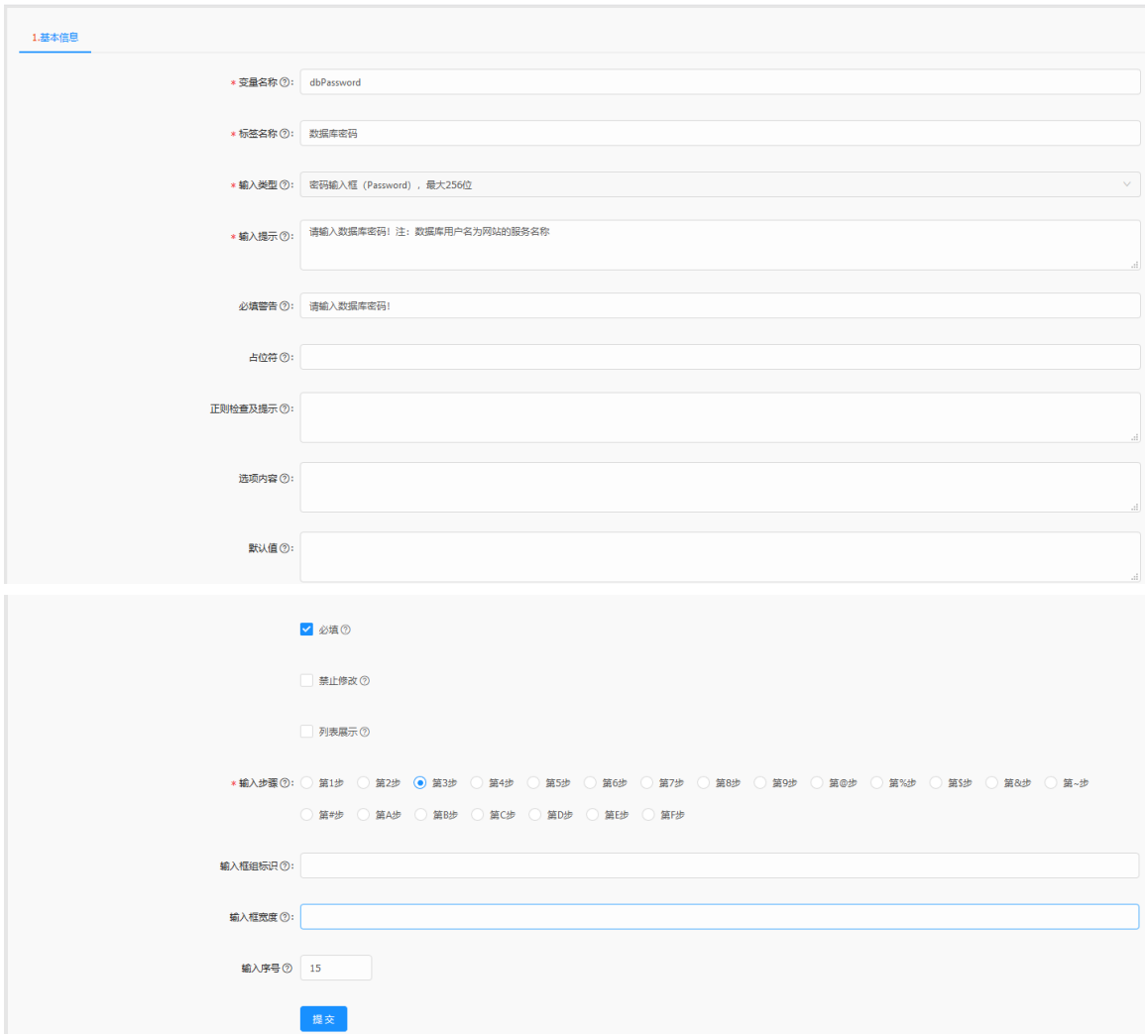


7.2 添加数据库的扩展变量：dbServiceId、dbPassword和dbCharset。如图：

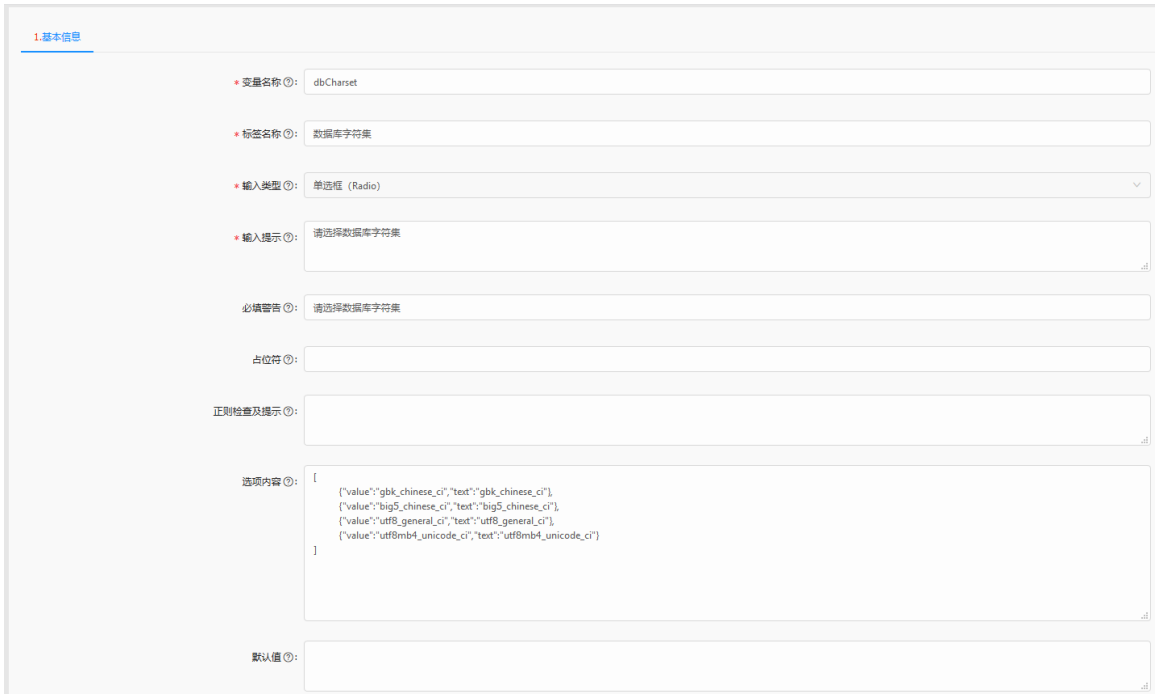


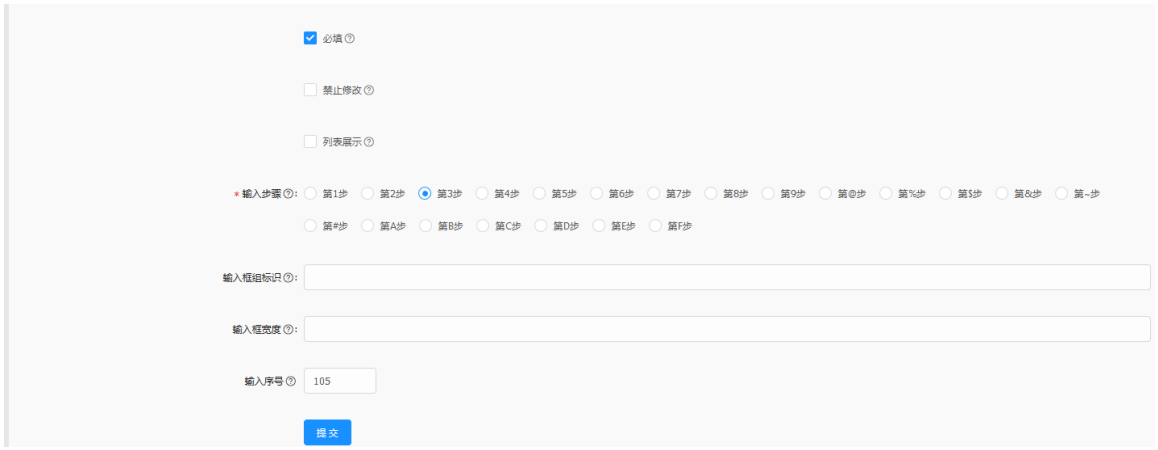


添加数据库密码扩展变量：dbPassword



添加数据库字符集扩展变量：dbCharset




必填
禁止修改
列表展示
*输入步骤:
第1步
第2步
第3步
第4步
第5步
第6步
第7步
第8步
第9步
第@步
第%步
第$步
第&步
第~步
第#步
第A步
第B步
第C步
第D步
第E步
第F步
输入框组标识:
输入框宽度:
输入序号
105
提交