Holoscan CLI - Package Command

Table of contents

`holoscan package` - generate HAP-compliant container for your application.

Synopsis

`holoscan package` [--help|-h] [--log-level|-l {DEBUG,INFO,WARN,ERROR,CRITICAL}] --config|-c CONFIG [--docs|-d DOCS] [--models|-m MODELS] --platform PLATFORM [--platform-config PLATFORM_CONFIG] [--timeout TIMEOUT] [--version VERSION] [--base-image BASE_IMAGE] [--build-image BUILD_IMAGE] [--build-cache BUILD_CACHE] [--cmake-args CMAKE_ARGS] [--no-cache|-n] [--sdk SDK] [--sdk-version SDK_VERSION] [--holoscan-sdk-file HOLOSCAN_SDK_FILE] [--monai-deploy-sdk-file MONAI_DEPLOY_SDK_FILE] [--output|-o OUTPUT] --tag|-t TAG [--username USERNAME] [--uid UID] [--gid GID] application

Examples

The code below package a python application for x86_64 systems:

> *# Using a Python directory as input # Required: a `__main__.py` file in the application directory to execute # Optional: a `requirements.txt` file in the application directory to install dependencies* holoscan package --platform x64-workstation --tag my-awesome-app --config /path/to/my/awesome/application/config.yaml /path/to/my/awesome/application/ *# Using a Python file as input* holoscan package --platform x64-workstation --tag my-awesome-app --config /path/to/my/awesome/application/config.yaml /path/to/my/awesome/application/my-app.py

The code below package a C++ application for the IGX Orin DevKit (aarch64) with a discrete GPU:

> *# Using a C++ source directory as input # Required: a `CMakeLists.txt` file in the application directory* holoscan package --platform igx-orin-devkit --platform-config dgpu --tag my-awesome-app --config /path/to/my/awesome/application/config.yaml /path/to/my/awesome/application/ *# Using a C++ pre-compiled executable as input* holoscan package --platform igx-orin-devkit --platform-config dgpu --tag my-

```
awesome-app --config /path/to/my/awesome/application/config.yaml
/path/to/my/awesome/bin/application-executable
```

> **Note**
>
> The commands above load the generated image onto Docker to make the image accessible with `docker images`.
>
> If you need to package for a different platform or want to transfer the generated image to another system, use the `--output /path/to/output` flag so the generated package can be saved to the specified location.

Positional Arguments

`application`

Path to the application to be packaged. The following inputs are supported:

- **C++ source code**: you may pass a directory path with your C++ source code with a `CMakeLists.txt` file in it, and the **Packager** will attempt to build your application using CMake and include the compiled application in the final package.
- **C++ pre-compiled executable**: A pre-built executable binary file may be directly provided to the **Packager**.
- **Python application**: you may pass either:
  - a directory which includes a `__main__.py` file to execute (required) and an optional `requirements.txt` file that defined dependencies for your Python application, or
  - the path to a single python file to execute

> ⚠ **Warning**
>
> Python (PyPI) modules are installed into the user's (via [--username USERNAME] argument) directory with the user ID specified via [--uid UID]. Therefore, when running a packaged Holoscan application on Kubernetes or other service providers, running Docker with non root user, and running Holoscan CLI `run` command where the logged-on user's ID is different, ensure to specify the `USER ID` that is used when building the application package.
>
> For example, include the `securityContext` when running a Holoscan packaged application with `UID=1000` using Argo:
>
> ```
> spec: securityContext: runAsUser: 1000 runAsNonRoot: true
> ```

Flags

`--config|-c CONFIG`

Path to the application's configuration file. The configuration file must be in `YAML` format with a `.yaml` file extension.

`[--docs|-d DOCS]`

An optional directory path of documentation, README, licenses that shall be included in the package.

`[--models|-m MODELS]`

An optional directory path to a model file, a directory with a single model, or a directory with multiple models.

Single model example:

```
my-model/        surgical_video.gxf_entities        surgical_video.gxf_index my-model/
    model        surgical_video.gxf_entities        surgical_video.gxf_index
```

Multi-model example:

```
my-models/       model-1          my-first-model.gxf_entities          my-first-
model.gxf_index       model-2       my-other-model.ts
```

`--platform PLATFORM`

A comma-separated list of platform types to generate. Each platform value specified generates a standalone container image. If you are running the **Packager** on the same architecture, the generated image is automatically loaded onto Docker and is available with `docker images`. Otherwise, use `--output` flag to save the generated image onto the disk.

`PLATFORM` must be one of: `clara-agx-devkit`, `igx-orin-devkit`, `jetson-agx-orin-devkit`, `x64-workstation`.

- `igx-orin-devkit`: IGX Orin DevKit
- `jetson-agx-orin-devkit`: Orin AGX DevKit
- `x64-workstation`: systems with a x86-64 processor(s)

`[--platform-config PLATFORM_CONFIG]`

Specifies the platform configuration to generate. `PLATFORM_CONFIG` must be one of: `igpu`, `igpu-assist`, `dgpu`.

- `igpu`: Supports integrated GPU
- `igpu-assist`: Supports compute-only tasks on iGPU in presence of a dGPU
- `dgpu`: Supports dedicated GPU

> **Note**
>
> `--platform-config` is required when `--platform` is not `x64-workstation` (which uses `dgpu`).

[--timeout TIMEOUT]

An optional timeout value of the application for the supported orchestrators to manage the application's lifecycle. Defaults to `0`.

[--version VERSION]

An optional version number of the application. When specified, it overrides the value specified in the configuration file.

[--base-image BASE_IMAGE]

Optionally specifies the base container image for building packaged application. It must be a valid Docker image tag either accessible online or via `docker images. By default, the **Packager** picks a base image to use from NGC.

[--build-image BUILD_IMAGE]

Optionally specifies the build container image for building C++ applications. It must be a valid Docker image tag either accessible online or via `docker images. By default, the **Packager** picks a build image to use from NGC.

[--build-cache BUILD_CACHE]

Specifies a directory path for storing Docker cache. Defaults to `~/.holoscan_build_cache`. If the `$HOME` directory is inaccessible, the CLI uses the `/tmp` directory.

[--cmake-args CMAKE_ARGS]

A comma-separated list of *cmake* arguments to be used when building C++ applications.

For example:

```
holoscan package --cmake-args "-DCMAKE_BUILD_TYPE=DEBUG -DCMAKE_ARG=VALUE"
```

[--no-cache|-n]

Do not use cache when building image.

[--sdk SDK]

SDK for building the application: Holoscan or MONAI-Deploy. `SDK` must be one of: holoscan, monai-deploy.

[--source URL|FILE]

Override the artifact manifest source with a securely hosted file or from the local file system.

E.g. https://my.domain.com/my-file.json

[--sdk-version SDK_VERSION]

Set the version of the SDK that is used to build and package the Application. If not specified, the packager attempts to detect the installed version.

[--holoscan-sdk-file HOLOSCAN_SDK_FILE]

Path to the Holoscan SDK Debian or PyPI package. If not specified, the packager downloads the SDK file from the internet depending on the SDK version detected/specified. The `HOLOSCAN_SDK_FILE` filename must have `.deb` or `.whl` file extension for Debian package or PyPI wheel package, respectively.

[--monai-deploy-sdk-file MONAI_DEPLOY_SDK_FILE]

Path to the MONAI Deploy App SDK Debian or PyPI package. If not specified, the packager downloads the SDK file from the internet based on the SDK version. The `MONAI_DEPLOY_SDK_FILE` package filename must have `.whl` or `.gz` file extension.

[--output|-o OUTPUT]

Output directory where result images will be written.

> **Note**
>
> If this flag isn't present, the packager will load the generated image onto Docker to make the image accessible with `docker images`. The `--output` flag is therefore required when building a packaging for a different target architecture than the host system that runs the packaer.

--tag|-t TAG

Name and optionally a tag (format: `name:tag`).

For example:

```
my-company/my-application:latest my-company/my-application:1.0.0 my-application:1.0.1 my-application
```

[--username USERNAME]

Optional *username* to be created in the container execution context. Defaults to `holoscan`.

[--uid UID]

Optional *user ID* to be associated with the user created with `--username` with default of `1000`.

> **Warning**

A very large UID value may result in a very large image due to an open issue with Docker. It is recommended to use the default value of `1000` when packaging an application and use your current UID/GID when running the application.

[--gid GID]

Optional *group ID* to be associated with the user created with `--username` with default of `1000`.

[--source PATH|URL]

Overrides the default manifest file source. This value can be a local file path or a HTTPS url.