API Builder CLI

This document will walk you through the process of how to publish API Builder apps to a newly deployed cluster using Appcelerator CLI commands.

The use of small containers should be avoided when publishing API Builder applications using the Appcelerator CLI commands. The minimum recommended container size for Arrow Apps is "Medium". Though you may be able to deploy to a "Dev" or "Small" container, for better memory usage and performance it is highly recommended that you use medium or bigger size containers.

Setup

You will need Appcelerator CLI (appc cli) version 7.0.0+. To install Appcelerator CLI 7.0.0, execute this command:

```
npm install appcelerator -g
```

Publish source code

The legacy method to publish the API Builder app is as follows:

1. Create a new API Builder app:

   ```
   $ appc new -t arrow
   Appcelerator Command-Line Interface, version 7.0.0
   Copyright (c) 2014-2017, Appcelerator, Inc.  All Rights Reserved.

   ? What's the project name? appctest
   *** new completed. ***
   ```

2. Update `appc.json` environment to add `extra_hosts` if you haven't set up a DNS yet. If your cluster uses `appctest.com` as a domain name, then you need to add `NODE_ENV` as well:

```
$ cd appctest
$ vi appc.json
{
...
  "cloud": {
    ...
    // Change your container size here
    "container": "Medium",
    // Number of containers for your app
    "minimum": 3,
    // Maximum number of containers for you app when autoscalling kicks in
    "maximum": 5,
    // NODE_ENV is only needed if
    // a. You are running in an alternately-named environment (like staging)
    // b. You are running on a domain with appctest in the name
    "environment": {"NODE_ENV":"PRODUCTION","extra_hosts":"54.212.208.81
api.cloudapp-1.appctest.com"},
    ...
   }
}
```

3. Execute `appc install` (required prior to the release of version 6.2.0).
4. Then publish the app normally by executing this command: `appc publish`

Publish by providing Dockerfile

Starting from AMPLIFY Runtime Services (formerly known as Arrow Cloud) 1.6.0, you can publish to AMPLIFY Runtime Services by pushing a Docker image directly. If your cluster has no Internet access, this is the recommended method to publish.

1. Create a new API Builder app:

```
$ appc new -t arrow
Appcelerator Command-Line Interface, version 7.0.0
Copyright (c) 2014-2017, Appcelerator, Inc.  All Rights Reserved.

? What's the project name? appctest
*** new completed. ***
```

2. Prepare a Dockerfile by creating a Dockerfile under your project directory and use the `start_app` script. Use of `start_app` is required if you want to able to retrieve access and console logs of your app using the `appc cloud logcat` and `appc cloud accesslog` commands. The following is a sample Dockerfile for an API Builder app:

```
FROM mhart/alpine-node:5.12.0
# Install curl command for app health check
RUN apk add --no-cache curl

# This is the script to start app
COPY start_app /usr/local/bin/
RUN chmod 755 /usr/local/bin/start_app

RUN mkdir -p /opt/app
WORKDIR /opt/app

COPY package.json /opt/app
COPY app.js /opt/app
COPY README.md /opt/app

RUN npm install


ENTRYPOINT ["/usr/local/bin/start_app"]
```

3. The `start_app` script should look something like this and be placed under the project directory with the Dockerfile:

```
Expand source
#!/bin/sh

APP_DIR="/opt/app"

# command to retrieve the containerId inside docker container
CONTAINID=$(cat /proc/1/cgroup | grep 'docker/' | tail -1 | sed 's/^.*\///' | cut
-c 1-12)

# replace the fake "serverId" with the real container id
if [ ! -z $CONTAINID ]; then
 ARROWCLOUD_APP_LOG_DIR=$(echo ${ARROWCLOUD_APP_LOG_DIR} | sed
"s/serverId/${CONTAINID}/")
 export serverId=${CONTAINID}
fi

APP_LOG_DIR="/ctdebuglog/${ARROWCLOUD_APP_LOG_DIR}"
APP_DEBUG_LOG_DIR="${APP_LOG_DIR}/debug"
APP_REQUESTS_LOG_DIR="${APP_LOG_DIR}/requests"

mkdir -p "${APP_DEBUG_LOG_DIR}"
if [ $? -ne 0 ]; then
 echo "Failed to create ${APP_DEBUG_LOG_DIR}"
 exit 1
fi

mkdir -p "${APP_REQUESTS_LOG_DIR}"
if [ $? -ne 0 ]; then
 echo "Failed to create ${APP_REQUESTS_LOG_DIR}"
 exit 1
fi
```

```
# make a symbolic link from ${APP_REQUESTS_LOG_DIR} to /ctlog to satisfy
appc-logger
ln -s ${APP_REQUESTS_LOG_DIR} /ctlog
if [ $? -ne 0 ]; then
 echo "Failed to create link from ${APP_REQUESTS_LOG_DIR} to /ctlog"
 exit 1
fi

STDOUT_LOG_FILE="${APP_DEBUG_LOG_DIR}/stdout.log"
STDERR_LOG_FILE="${APP_DEBUG_LOG_DIR}/stderr.log"

# curl is used for health-check when creating docker service for an app.
curl >/dev/null 2>&1
if [ $? -eq 127 ]; then
 echo "curl not found! App image must include curl for health-check purpose." >>
${STDOUT_LOG_FILE} 2>>${STDERR_LOG_FILE}
 exit 1
fi

cd $APP_DIR
echo "[app_launcher] starting application via \"$DAEMON $DAEMON_ARGS\"" >>
${STDOUT_LOG_FILE} 2>>${STDERR_LOG_FILE}
# start the app in the same process which is the main process(pid 1), so that the
app can get signals
exec node app.js >> ${STDOUT_LOG_FILE} 2>>${STDERR_LOG_FILE}
RETVAL=$?
if [ $RETVAL -ne 0 ]; then
 echo "[app_launcher] application is over with status code $RETVAL." >>
${STDOUT_LOG_FILE} 2>>${STDERR_LOG_FILE}
else
 echo "[app_launcher] application is over." >> ${STDOUT_LOG_FILE}
```

```
2>>${STDERR_LOG_FILE}
fi
exit $RETVAL
```

4. Execute `appc publish`. You will need to provide the app version using the `--app-version` flag. You should always provide the app name since Appcelerator CLI will not scan `package.json` as it would normally with source code publishing that does obtain the app name. Using `appc publish` will build the Docker image by using the provided Dockerfile and push the image to AMPLIFY Runtime Services directly:

```
$ cd appctest
$ appc publish --app-version 1.0.0 appctest
```

5. If you need to scale up the number of servers, execute these commands:

```
# Set maximum number of containers allowed for appctest
$ appc cloud config appctest --maxsize <size>
# Set current number of containers for appctest as long as the cluster has enough
resource)
$ appc cloud config appcteest --minsize <size>
```

Publish by using existing Docker image

Alternatively, after preparing your Dockerfile and `start_app` script, you can build the Docker image yourself and publish the image to AMPLIFY Runtime Services:

1. Build the Docker image:

```
$ cd appctest
$ docker build -t appctestimage .
# Double check the image presents locally
$ docker images |grep appctestimage
appctestimage               latest              869918dab71b        43 minutes
ago      277 MB
```

2. Next, publish the image directly. Ensure that you provide the app version and name and image name by using the `--app-version`, `app name`, and `--image` flags:

```
$ cd appctest
$ appc publish --app-version 1.0.0 --image appctestimage appctest
```

3. If you need to scale up the number of servers, execute these commands:

```
# Set maximum number of containers allowed for appctest
$ appc cloud config appctest --maxsize <size>
# Set current number of containers for appctest as long as the cluster has enough
resource)
$ appc cloud config appcteest --minsize <size>
```


If you haven't set up your DNS yet, your app publish may fail with the following error:

```
$ appc cloud logcat appctest
...
Uncaught Exception Error loading connector/appc.arrowdb. Error:
getaddrinfo ENOTFOUND api.cloudapp-1.appctest.com
api.cloudapp-1.appctest.com:443
2017-04-03T16:24:39-07:00 | Error: Error loading connector/appc.arrowdb.
Error: getaddrinfo ENOTFOUND api.cloudapp-1.appctest.com
api.cloudapp-1.appctest.com:443
...
```

In this case, you will need to execute the following command to configure the custom host info in the app container. Please note if you try to update `/etc/hosts` file in the Dockerfile with the custom hostname and IP. It will not work because the Docker swarm mode will override that information at the time of the container launch.

```
$ appc cloud config --set "extra_hosts=54.212.208.81
api.cloudapp-1.appctest.com"
Appcelerator Command-Line Interface, version 7.0.0
Copyright (c) 2014-2017, Appcelerator, Inc.  All Rights Reserved.
Admin Hostname: https://admin.cloudapp-1.appctest.com
The variable has been saved successfully.

# Confirm the env is set correctly
$ appc cloud config --env appctest
Appcelerator Command-Line Interface, version 7.0.0
Copyright (c) 2014-2017, Appcelerator, Inc.  All Rights Reserved.
Admin Hostname: https://admin.cloudapp-1.appctest.com
extra_hosts = 54.212.208.81 api.cloudapp-1.appctest.com
```

⚠ Note about Docker image publish with Alpine: You will need to execute `apk add --no-cache curl` in the Docker file when publishing the Docker image.

CLI commands

Command	Description
`appc login`	Login
`appc config set defaultEnvironment preproduction`	Selection of an environment (optional, if the default needs to be switched).
`appc new`	Create an app.
`appc publish`	Publish the app.
`appc cloud list`	Query the list of applications that are deployed to my current environment and dashboard.
`appc cloud list <appname>`	Query the config for a specific application that is deployed to the currently referenced environment and dashboard.
`appc cloud config --maxsize 4 <appname>`	Configuration for a given application: change the maximum number of containers.
`appc cloud config –minsize 2 <appname>`	Configuration for a given application: change the minimum number of containers.

`appc cloud server <appname> –set <value>`	Change the size of the given application. Accepted values include `Dev`, `Small`, `Medium`, `Large`, and `XLarge`.
`appc cloud logcat`	View logs of the application.
`appc cloud loglist –build_logs`	View logs of the application.
`appc cloud accesslog`	List application's access log.
`appc cloud add`	Add a new route or service.
`appc cloud cname`	Set a CNAME for an application.
`appc cloud config`	Configure the application.
`appc cloud crt`	Manage SSL certificates for the application.
`appc cloud download`	Download source files for the specified application and version. ⚠ If you publish using a Docker image, this command won't work.
`appc logout`	Log out
`appc remove`	Removes installed Appcelerator CLI.
`appc run`	Run an application locally for dev. and/or testing.
`appc unpublish`	Unpublish an application.
`appc cloud whoami`	Show current login user.