Initial Steps in Porting Redis:

1. Make a directory that is meant specifically for you.
    a. Ex: mkdir aquiles

2. Go into that directory
    a. cd aquiles

3. Use the command “zopen generate” to specify the tool
    a. How it is formatted:
        i. Project Name
        ii. Description
        iii. License
        iv. Source URLs
            1. Grab the SSH link from the project
        v. Build Dependencies
            1. In the READ.me, the build dependencies should be listed
        vi. Runtime Dependencies
            1. In the READ.me, the runtime dependencies should also be listed

4. CD into your project name
    a. Ex: cd redisport

5. Go into the buildenv file using a text editor
    a. This could be done with an editor outside of the UNIX system such as VS Studio, but it could also be done with Vim or nano.
6. Ensure the information there aligns with the build information from the official redis github.
    a. The buildenv file defines the environment in which the project will be built and tested. Here are some aspects of this file:
        i. Dependencies: Lists the libraries and tools necessary to build the project. May have specified specific versions.
            1. How it may look:
                a. export ZOPEN_STABLE_DEPS="make openssl which pkgconfig check_python"

        ii. Environment variables: Sets the variables needed for the build process like paths to tools and libraries

1. export ZOPEN_STABLE_URL="https://github.com/redis/redis.git"

iii. Build tools: Specifies the tools and compilers used for building the project
   1. export ZOPEN_COMP="CLANG"

iv. Scripts: Automate the process of building
   1. # bump: redis-version /REDIS_VERSION="(.*)"/ https://github.com/redis/redis.git|semver:*
   2. # REDIS_VERSION="V.R.M" # Specify a stable release

v. Configurations Settings: Contains settings that configure the build process such as optimization flags and compiler options
   1. zopen_pre_patch()
   2. {
   3. export CFLAGS="$CFLAGS $CPPFLAGS -mzos-target=zosv2r5 -D__XPLAT -D_OPEN_THREADS=2 -D__thread= -DMAP_ANON=0"
   4. }

7. Save changes then exit out of buildenv
   a. Use ctrl+shift+o to save the changes made to the file
   b. Use ctrl+shift+x for exiting the file

8. Build the project using zopen build -vv
   a. -vv will allow for more detailed information about how the project is building

Commiting changes to Github:

1. Cd into the main repository
   a. Ex: cd aquiles/redisport/redis

2. Use git diff to take the changes made and put in your patch
   a. Ex: git diff > ../patches/pr.patch

3. Create a new patch that has all the new changes:
   a. Ex: git add buildenv patches/pr1.patch

4. (Optional) You could remove a previous patch:
    a. Ex: rm patches/PR1.patch

5. Specify the SSH key you are using to get ready to commit:
    a. Ex: export GIT_SSH_COMMAND="what you have as the ssh"

6. Commit your changes using “git commit”
    a. This will take you to a screen asking you to add a comment for that commit.
    b. Once you are done press escape to get out of insert mode
    c. From there press “:” by doing Shift+; which will put you into command mode
    d. Type wq to save then quit.
        i. This way the commit goes through

7. (Optional) Fix the SSH authorization error since same user is shared:
    a. First use this: export GIT_SSH_COMMAND="/bin what you have as the ssh"
    b. export GIT_TRACE=1

8. Push it into the main repository using “git push”