Overview

In this artifact, we included the implementation of our web automation program synthesizer called Arborist. We will evaluate its efficiency in synthesizing programs for challenging web automation tasks and compare it with state-of-the-art techniques. In the following, we first present benchmarks we used in our evaluation, then show instructions to reproduce our experiment.

Benchmarks

We collected 131 real-world benchmarks on web-automation tasks, with each benchmark stored in a folder with name WxTx. Each benchmark contains the following files/folder:

- *.program: the ground truth web automation program.
- script.py: the selenium program used to record the action trace and doms of the simulated user interactions on the doms.
- trace.json: the json file contains a list of actions for completing the benchmark task. Each action contains its type, corresponding selectors, and the data if applicable.
- doms/: the folder contain all website dom indexed from the starting website dom before applying the first action to the last website dom after applying the last action.

Evaluation Instructions

We packaged all code and data into a cross-platform Docker container that works for both x86 and arm host machines.

Setup Docker Desktop

1. Install Docker Desktop if you haven't already from https://www.docker.com/products/docker-desktop/
2. Change the allocated disk space to **200GB** in the preferences (Settings -> Resources -> Disk). Docker won't necessarily use all 200GB but this amount is required to prevent out of memory errors. See Figure 1 on the next page for the steps.

Run main experiment

1. Pull and run the docker container for arborist:

```
docker run -ti alienkevin/arborist
```

2. Unzip test folder

```
mkdir tests
tar -I pigz -xf tests.tar.gz
```

Ignore warnings like below:

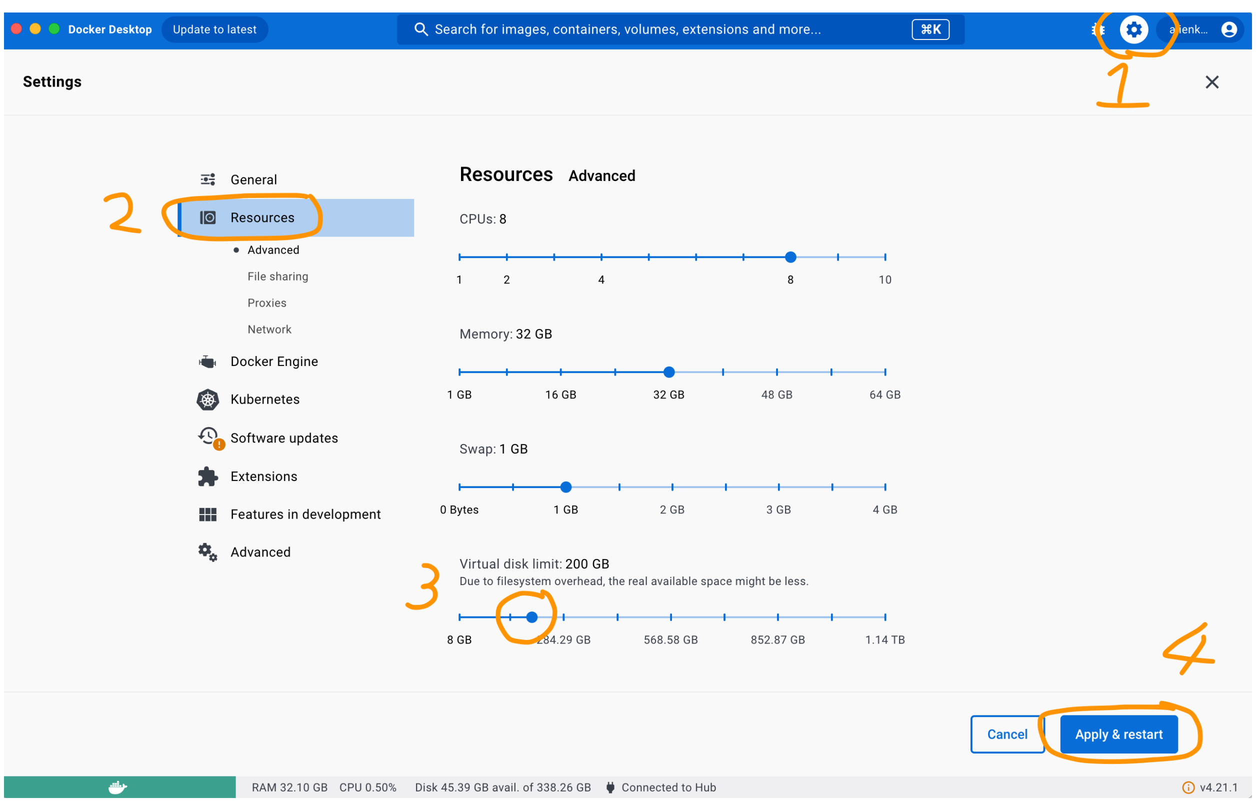
```
tar: Ignoring unknown extended header keyword 'SCHILY.fflags'
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.FinderInfo'
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.lastuseddate#PS'
...
```

3. Run the main experiment

```
make main
```

The main experiment takes around 40 minutes wall-clock time on an M1 Max MacBook Pro.

4. Plot the experiment results



Resources Advanced

CPUs: 8

Memory: 32 GB

Swap: 1 GB

Virtual disk limit: 200 GB
Due to filesystem overhead, the real available space might be less.

RAM 32.10 GB CPU 0.50% Disk 45.39 GB avail. of 338.26 GB Connected to Hub v4.21.1

Figure 1: Docker Desktop Configuration

```
python plot.py
```

5. Export results to host machine. (Below steps all happens on your host machine's terminal, not the container's terminal)

   a. First get the name of the container running under the NAMES column:

```
$ docker ps
CONTAINER ID   IMAGE      COMMAND       CREATED          STATUS          PORTS     NAMES
e705b63f2247   arborist   "/bin/bash"   37 minutes ago   Up 37 minutes             peaceful_hertz
```

Here the name is `peaceful_hertz`.

   b. Then, copy the generated figures from the container to your host machine. Here, we show how to copy the figures to the Downloads folder.

```
docker cp peaceful_hertz:/workspace/figures ~/Downloads/figures
```

You should see that the two figures look similar to Figure 16 on page 19 of the paper.

   c. You can also copy the generated result spread sheets to your host machine for inspection.

```
docker cp peaceful_hertz:/workspace/benchmark_summary.csv ~/Downloads/benchmark_summary.csv
```