Troubleshooting

Chrome headless doesn't launch

Make sure all the necessary dependencies are installed. You can run **`ldd chrome | grep not`** on a Linux machine to check which dependencies are missing. The common ones are provided below.

▶ Debian (e.g. Ubuntu) Dependencies

▶ CentOS Dependencies

- Check out discussions:
- #290 - Debian troubleshooting
- #391 - CentOS troubleshooting
- #379 - Alpine troubleshooting

Chrome Headless fails due to sandbox issues

- Make sure kernel version is up-to-date.
- Read about linux sandbox here: https://chromium.googlesource.com/chromium/src/+/master/docs/linux_suid_sandbox_development.md
- Try running without the sandbox (**Note: running without the sandbox is not recommended due to security reasons!**)

```
const browser = await puppeteer.launch({args: ['--no-sandbox', '--disable-setuid-sandbox']});
```

Running Puppeteer on Travis CI

To run headless Chrome on Travis, you *must* call **`launch()`** with flags to disable Chrome's sandbox, like so:

```
const browser = await puppeteer.launch({args: ['--no-sandbox']});
```

Running Puppeteer in Docker

Getting headless Chrome up and running in Docker can be tricky. The bundled Chromium that Puppeteer installs is missing the necessary shared library dependencies.

To fix, you'll need to install the missing dependencies and the latest Chromium package in your Dockerfile:

```
FROM node:8-slim

# See https://crbug.com/795759
RUN apt-get update && apt-get install -yq libgconf-2-4
```

```
# Install latest chrome dev package and fonts to support major charsets (Chinese, Japanese, Arabic,
# Note: this installs the necessary libs to make the bundled version of Chromium that Puppeteer
# installs, work.
RUN apt-get update && apt-get install -y wget --no-install-recommends \
    && wget -q -O - https://dl-ssl.google.com/linux/linux_signing_key.pub | apt-key add - \
    && sh -c 'echo "deb [arch=amd64] http://dl.google.com/linux/chrome/deb/ stable main" >> /etc/ap
    && apt-get update \
    && apt-get install -y google-chrome-unstable fonts-ipafont-gothic fonts-wqy-zenhei fonts-thai-t
      --no-install-recommends \
    && rm -rf /var/lib/apt/lists/* \
    && apt-get purge --auto-remove -y curl \
    && rm -rf /src/*.deb

# It's a good idea to use dumb-init to help prevent zombie chrome processes.
ADD https://github.com/Yelp/dumb-init/releases/download/v1.2.0/dumb-init_1.2.0_amd64 /usr/local/bir
RUN chmod +x /usr/local/bin/dumb-init

# Uncomment to skip the chromium download when installing puppeteer. If you do,
# you'll need to launch puppeteer with:
#     browser.launch({executablePath: 'google-chrome-unstable'})
# ENV PUPPETEER_SKIP_CHROMIUM_DOWNLOAD true

# Install puppeteer so it's available in the container.
RUN npm i puppeteer

# Add user so we don't need --no-sandbox.
RUN groupadd -r pptruser && useradd -r -g pptruser -G audio,video pptruser \
    && mkdir -p /home/pptruser/Downloads \
    && chown -R pptruser:pptruser /home/pptruser \
    && chown -R pptruser:pptruser /node_modules

# Run everything after as non-privileged user.
USER pptruser

ENTRYPOINT ["dumb-init", "--"]
CMD ["google-chrome-unstable"]
```

Build the container:

```
docker build -t puppeteer-chrome-linux .
```

Run the container by passing **`node -e "<yourscript.js content as a string>`** as the command:

```
 docker run -i --rm --cap-add=SYS_ADMIN \
   --name puppeteer-chrome puppeteer-chrome-linux \
   node -e "`cat yourscript.js`"
```

There's a full example at https://github.com/ebidel/try-puppeteer that shows how to run this Dockerfile from a webserver running on App Engine Flex (Node).

Running on Alpine

The newest Chromium package supported on Alpine is 64, which was corresponding to Puppeteer v0.13.0.

Example Dockerfile:

```
FROM node:9-alpine

# Installs latest Chromium (64) package.
RUN apk update && apk upgrade && \
    echo @edge http://nl.alpinelinux.org/alpine/edge/community >> /etc/apk/repositories && \
    echo @edge http://nl.alpinelinux.org/alpine/edge/main >> /etc/apk/repositories && \
    apk add --no-cache \
      chromium@edge \
      nss@edge

...

# Tell Puppeteer to skip installing Chrome. We'll be using the installed package.
ENV PUPPETEER_SKIP_CHROMIUM_DOWNLOAD true

# Puppeteer v0.13.0 works with Chromium 64.
RUN yarn add puppeteer@0.13.0

# Add user so we don't need --no-sandbox.
RUN addgroup -S pptruser && adduser -S -g pptruser pptruser \
    && mkdir -p /home/pptruser/Downloads \
    && chown -R pptruser:pptruser /home/pptruser \
    && chown -R pptruser:pptruser /app

# Run everything after as non-privileged user.
USER pptruser

...
```

And when launching Chrome, be sure to use the `chromium-browser` executable:

```
const browser = await puppeteer.launch({
  executablePath: '/usr/bin/chromium-browser'
});
```

Tips

By default, Docker runs a container with a `/dev/shm` shared memory space 64MB. This is typically too small for Chrome and will cause Chrome to crash when rendering large pages. To fix, run the container with `docker run --shm-size=1gb` to increase the size of `/dev/shm`. Since Chrome 65, this is no longer necessary. Instead, launch the browser with the `--disable-dev-shm-usage` flag:

```
const browser = await puppeteer.launch({
  args: ['--disable-dev-shm-usage']
});
```

This will write shared memory files into `/tmp` instead of `/dev/shm`. See crbug.com/736452 for more details.

Seeing other weird errors when launching Chrome? Try running your container with `docker run --cap-add=SYS_ADMIN` when developing locally. Since the Dockerfile adds a `pptr` user as a non-privileged user, it may not have all the necessary privileges.

dumb-init is worth checking out if you're experiencing a lot of zombies Chrome processes sticking around. There's special treatment for processes with PID=1, which makes it hard to terminate Chrome properly in some cases (e.g. in Docker).

Running Puppeteer on Heroku

Running Puppeteer on Heroku requires some additional dependencies that aren't included on the Linux box that Heroku spins up for you. To add the dependencies on deploy, add the Puppeteer Heroku buildpack to the list of buildpacks for your app under Settings > Buildpacks.

The url for the buildpack is https://github.com/jontewks/puppeteer-heroku-buildpack

When you click add buildpack, simply paste that url into the input, and click save. On the next deploy, your app will also install the dependencies that Puppeteer needs to run.

If you need to render Chinese, Japanese, or Korean characters you may need to use a buildpack with additional font files like https://github.com/CoffeeAndCode/puppeteer-heroku-buildpack

There's also another simple guide from @timleland that includes a sample project: https://timleland.com/headless-chrome-on-heroku/.

Running Puppeteer on AWS Lambda

AWS Lambda limits deployment package sizes to ~50MB. This presents challenges for running headless Chrome (and therefore Puppeteer) on Lambda. The community has put together a few resources that work around the issues:

- https://github.com/adieuadieu/serverless-chrome/blob/master/docs/chrome.md (tracks the latest Chromium snapshots)
- https://github.com/universalbasket/aws-lambda-chrome
- https://github.com/Kikobeats/aws-lambda-chrome