Using `pandoc` with GitHub Actions

You can use pandoc, the **universal markup converter**, on GitHub Actions to convert documents.

GitHub Actions is an Infrastructure as a Service (IaaS) from GitHub, that allows you to automatically run code on GitHub's servers on every push (or a bunch of other GitHub events). For example, you can use GitHub Actions to convert some `file.md` to `file.pdf` (via LaTeX) and upload the results to a web host.

Using `docker://pandoc` Images Directly

You can now *directly* reference container actions on GitHub Actions. You do not *need* a separate GitHub Action.

If you need LaTeX (because you want to convert through to PDF), you should use the `docker://pandoc/latex` image. Otherwise, the smaller `docker://pandoc/core` will suffice.

It is a good idea to be explicit about the pandoc version you require, such as `docker://pandoc/core:2.9`. This way, any future breaking changes in pandoc will not affect your workflow. You can find out whatever the latest released docker image is on docker hub. You should avoid specifying *no* tag or the `latest` tag – these will float to the latest image and will expose your workflow to potentially breaking changes.

Simple Usage

You can use pandoc inside GitHub actions exactly as you would use it on the command line. The string passed to `args` gets appended to the `pandoc` command.

The below example is equivalent to running `pandoc --help`.

You can see it in action here.

```yaml
name: Simple Usage

on: push

jobs:
  convert_via_pandoc:
    runs-on: ubuntu-22.04
    steps:
      - uses: docker://pandoc/core:2.9
        with:
          args: "--help" # gets appended to pandoc command
```

Long Pandoc Calls

Remember that as per the GitHub Actions workflow syntax, "an `array of strings` is not supported by the `jobs.<job_id>.steps.with.args` parameter. Pandoc commands can sometimes get quite long and unwieldy, but you must pass them as a *single* string. If you want to break up the string over several lines, you can use YAML's block chomping indicator:

```yaml
name: Long Usage

on: push

jobs:
  convert_via_pandoc:
    runs-on: ubuntu-22.04
    steps:
      - run: echo "foo" > input.txt  # create an example file
      - uses: docker://pandoc/core:2.9
        with:
          args: >-  # allows you to break string into multiple lines
            --standalone
            --output=index.html
            input.txt
```

You can see it in action here.

Advanced Usage

You can also:

- create an output directory to compile into; makes it easier to deploy outputs.
- upload the output directory to GitHub's artifact storage; you can quickly download the results from your GitHub Actions tab in your repo.

Remember that wildcard substitution (say, `pandoc *.md`) or other shell features frequently used with pandoc do not work inside GitHub Actions yaml files `args:` fields. Only GitHub Actions context and expression syntax can be used here. If you want to make use of such shell features, you have to run that in a separate step in a `run` field and store the result in the GitHub actions context. The below workflow includes an example of how to do this to concatenate several input files.

You can see it in action (haha!) here.

```yaml
name: Advanced Usage

on: push
```

```yaml
jobs:
  convert_via_pandoc:
    runs-on: ubuntu-22.04
    steps:
      - uses: actions/checkout@v3

      - name: create file list
        id: files_list
        run: |
          echo "Lorem ipsum" > lorem_1.md  # create two example files
          echo "dolor sit amet" > lorem_2.md
          mkdir output  # create output dir
          # this will also include README.md
          echo "files=$(printf '"%s" ' *.md)" > $GITHUB_OUTPUT

      - uses: docker://pandoc/latex:2.9
        with:
          args: --output=output/result.pdf ${{ steps.files_list.outputs.files }}

      - uses: actions/upload-artifact@v3
        with:
          name: output
          path: output
```

Lorem ipsum

dolor sit amet