An easier way to install tools for GitHub Actions

This article was fetched from an rss feed(13/10/2022)

I want to show you how to use arkade in your GitHub Actions to get the typical kinds of tools you'll need to build code and run end-to-end tests. By the end of the post, I hope you'll see why developers are using arkade on their workstations and in CI instead of the more clunky alternatives.

> arkade has 3.1k GitHub Stars, 157 releases, over 110k downloads and over 72 contributors.

Let's start right at the beginning.

When I started the arkade project in 2019, it was all about making it easier to install Helm charts for Kubernetes clusters. In one instance, I was writing a tutorial for Civo, and needed 5000 words to install the boilerplate software to get a TLS certificate.

Arkade's job was to reduce these 5000 words to 5 lines of text, and it worked. That's where `arkade install` came about. It could convert lengthy, repetitive technical content into concise steps:

```
arkade install ingress-nginx
arkade install cert-manager
arkade install openfaas
arkade install openfaas-ingress \
    --email alex@openfaas.com
    --domain faas.openfaas.com
```

How's that different from Helm? I've got a surprise for you. It is Helm, but wrapped with defaults so you can focus on your $dayjob instead of flicking through values.yaml files. Every "app" for Kubernetes has flags that we've hand-picked, see `arkade install openfaas --help` for instance.

But please don't let me distract you, this post is about `arkade get`. A much faster alternative to brew, and a much more up to date alternative to `apt-get`.

Then, it quickly became necessary to have CLIs at hand on Windows, MacOS, Linux and even ARM hosts like a Raspberry Pi, `arkade get` was born.

Works on MacOS, Darwin M1 and Intel, Linux and Linux ARM:

```
arkade get kubectl
arkade get kubectl@v1.25.0
arkade get kubectl faas-cli helm inletsctl
```

Try it out for yourself, you can download arkade to your workstation here: arkade

Extending GitHub Actions

There are three types of GitHub Action, we'll look at two of them: composite and JavaScript. For all intents and purposes, a composite action is what you use when you want to write bash.

Here's an example I wrote to setup SSHD with the actor's SSH keys pre-installed: alexellis/setup-sshd-actor

Using the action looks the same as any other:

```
  - name: Setup SSH server for Actor
    uses: alexellis/setup-sshd-actor@master
```

JavaScript actions are a better fit when you need some application logic, but don't want to ship and host a separate binary to be run in a composite action.

My earliest was [alexellis/upload-assets] which is a GitHub Action to upload multiple assets to a release using a wildcard. I made it because GitHub's own didn't support wildcards, and I needed that everywhere and it was incredibly useful in the transition from Travis CI.

Here's how you use it:

```
      - name: Upload release binaries
        uses: alexellis/upload-assets@0.2.2
        env:
          GITHUB_TOKEN: $
```

```
        with:
          asset_paths: '["./bin/release-it*"]'
```

Square peg, round CI system

Installing arkade inside a GitHub Action is as simple as:

```
name: Install arkade
run: curl -sLS https://get.arkade.dev | sudo sh
```

But ardent fans of GitHub Actions wanted more.

So I sat down and wrote a composite action to install the binary in a similar way, here it is:

```
# Copyright (c) 2022 OpenFaaS Ltd
name: 'Install arkade'
description: 'Install arkade'
branding:
  icon: 'arrow-right-circle'
  color: 'gray-dark'
runs:
    using: 'composite'
    steps:
      - name: Install Install
        shell: bash
        id: download
        run: |
          if ! [ -x "$(command -v curl)" ]; then
            sudo apt update && sudo apt install -qy curl
          fi
          curl -sLS https://get.arkade.dev | sudo sh
          echo "PATH=$HOME/.arkade/bin:$PATH" >> $GITHUB_ENV
```

Now you can write this instead:

```
uses: alexellis/setup-arkade@v2
```

What's that I hear? You're not impressed?

I thought we could do better too.

So I thought how I could build an action for the `get` command. I'd seen a friend (Stefan Prodan) had a hard-coded list of tools (that were also available in arkade).

He'd used inputs to control which tools to install:

```
      - name: Run Kubernetes tools
        uses: stefanprodan/kube-tools@v1
        with:
          kubectl: 1.23.0
          kustomize: 4.4.1
          helm: 2.17.0
          helmv3: 3.7.2
          kubeseal: 0.16.0
          kubeval: v0.16.1
```

Unfortunately, that also meant hard-coding the list in his actions.yml file, and having to write new code for every tool he added.

```
inputs:
  command:
    description: 'command to run'
    required: true
  kubectl:
    description: 'kubectl version'
  kustomize:
    description: 'kustomize version'
```

So could I do better?

I took inspiration from his design and came up with a tool that executed `arkade get -o list` and then created a new YAML file with all the inputs printed out, a bit like Stefan's file.

Rather than his dozen tools, you get access to 106 tools, with more added every week:

```
arkade get -o list |wc -l
106
```

Check out the program I wrote in Go: to-inputs/main.go

This program gets run every night to repopulate the actions.yml file using a good ole trick called `envsubst`.

Here how envsubst works:

```
cat<<EOF > template.txt
# Inputs from arkade:

$INPUTS

EOF

INPUTS=$(arkade get -o list)
cat template.txt | envsubst > result.txt

head -n 5 result.txt

# Inputs from arkade:

argocd
argocd-autopilot
arkade
...
```

You can see the full version of the nightly GitHub Action here which regenerates the list of tools and commits back to the repository: https://github.com/alexellis/arkade-get/blob/master/.github/workflows/update-tools.yml

Here's what it looks like now:

```
    - uses: alexellis/setup-arkade@v1
    - uses: alexellis/arkade-get@master
      with:
        kubectl: v1.25.0
        helm: latest
        argocd: latest
        inletsctl: latest
        faas-cli: 0.14.10
    - name: check for faas-cli
      run: |
        faas-cli version
```

Wrapping up

I came here to show you some of what I learned creating both composite and JavaScript actions, and how to work around a limitation for custom actions, where you can only specify a static list of inputs.

Need more convincing?

Two year update: Building an Open Source Marketplace for Kubernetes

What if you love brew?

That's OK, you're allowed to try arkade from time to time, especially if you don't like waiting to download CLIs.

What if you're an arduent apt-get convert and can't be convinced otherwise?

That's OK too, if you like old really packages or waiting for PPAs to update.

What if there's already a GitHub action to install this one tool?

Then feel free to use it, but for the ones that are missing their own action, or where that becomes too much to manage, Arkade has 106 tools at the moment, with more being added every week and month.

What else does arkade do?

Glad you asked, check out the README, it's full of practical examples: arkade.dev

How do I contribute a CLI to arkade?

Check out a recent PR where we added flyctl: add flyctl tool #769

Want a full example?

Here's what it looks like to install K3s using K3sup and then go on to deploy OpenFaaS. It's an example from the actuated docs.

```
name: k3sup-tester

on: push
jobs:
  k3sup-tester:
    runs-on: actuated
    steps:
    - uses: alexellis/setup-arkade@v1
    - uses: alexellis/arkade-get@master
      with:
        kubectl: v1.25.0
        k3sup: latest
        faas-cli: latest

      - name: Install K3s with k3sup
        run: |
          mkdir -p $HOME/.kube/
          k3sup install --local --local-path $HOME/.kube/config
      - name: Wait until nodes ready
        run: |
          k3sup ready --quiet --kubeconfig $HOME/.kube/config --context default
      - name: Install OpenFaaS
        run: arkade install openfaas
```

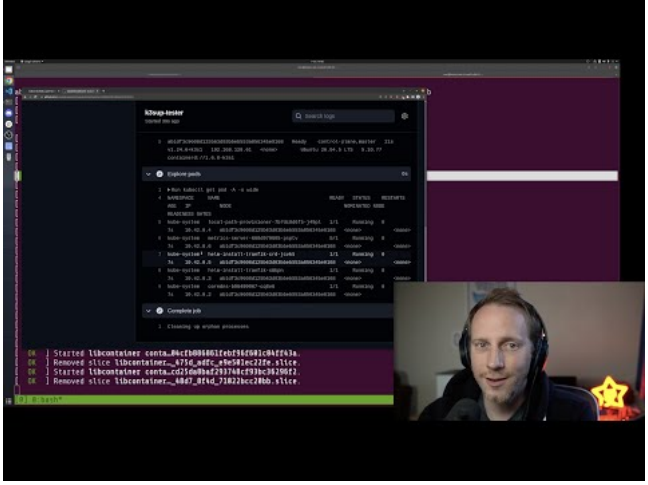
Whilst I've got you here

At OpenFaaS Ltd, we've been trying to make the experience for self-hosted runners faster, completely isolated and more efficient.

See if this resonates?

> GitHub Actions Managed/Hosted runners are great most of the time and super convenient
>
> but also:
>
> - limited on CPU/RAM
> - slower than your own hardware or cloud provider
> - limited to Intel runners
> - limited on concurrent jobs
> - and those e2e tests are making you miserable
>
> So..
>
> — Alex Ellis (@alexellisuk) September 24, 2022

Or watch the video:



Live demo and walk-through of actuated for secure self-hosted GitHub Actions