
CLOUDNATIVE
SECURITYCON
NORTH AMERICA 2023

CLOUDNATIVE SECURITYCON NORTH AMERICA 2023

Securing Self-Hosted GitHub Actions With Kubernetes and Actions-Runner-Controller

Natalie Somersall

Senior Solutions Engineer, GitHub

About me

Senior Solutions Engineer, GitHub

Love

- Empowering developers to change the world
- Doing cutting-edge stuff in highly regulated environments
- Automating *everything*

GitHub – @some-natalie

Web – https://some-natalie.dev

Where are we headed?

1. Why would I ever want to do this?
2. Kubernetes cluster settings
3. GitHub settings and deployment scopes
4. Multi-tenancy in actions-runner-controller
5. Runner images
6. Conclusions

All the normal container and Kubernetes security guidelines still apply!

I have a bias!

Friction is the largest driver of admins, users, and developers doing insecure things.

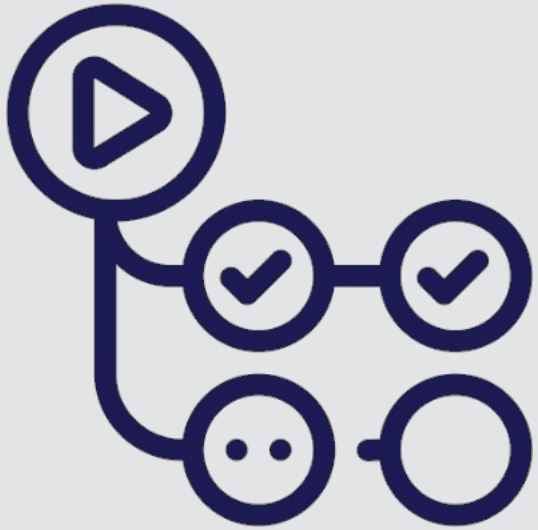
What's GitHub Actions?

GitHub's native automation platform, frequently used for **CI/CD** and running most of GitHub

- **Reusable, modular workflows** from an open-source marketplace or your internal organization

- Cloud folks can use our **hosted runners as a SaaS**
  - **Ephemeral** VMs
  - Lots and lots of dependencies pre-loaded (github.com/actions/runner-images)
  - **SBOM readily available** for each release
  - Updated (~weekly), automatic scaling, caching, etc. all handled for you

- **Bring your own compute!** (free-ish)
  - Needs the github.com/actions/runner agent
  - **Security operations is all on your team**

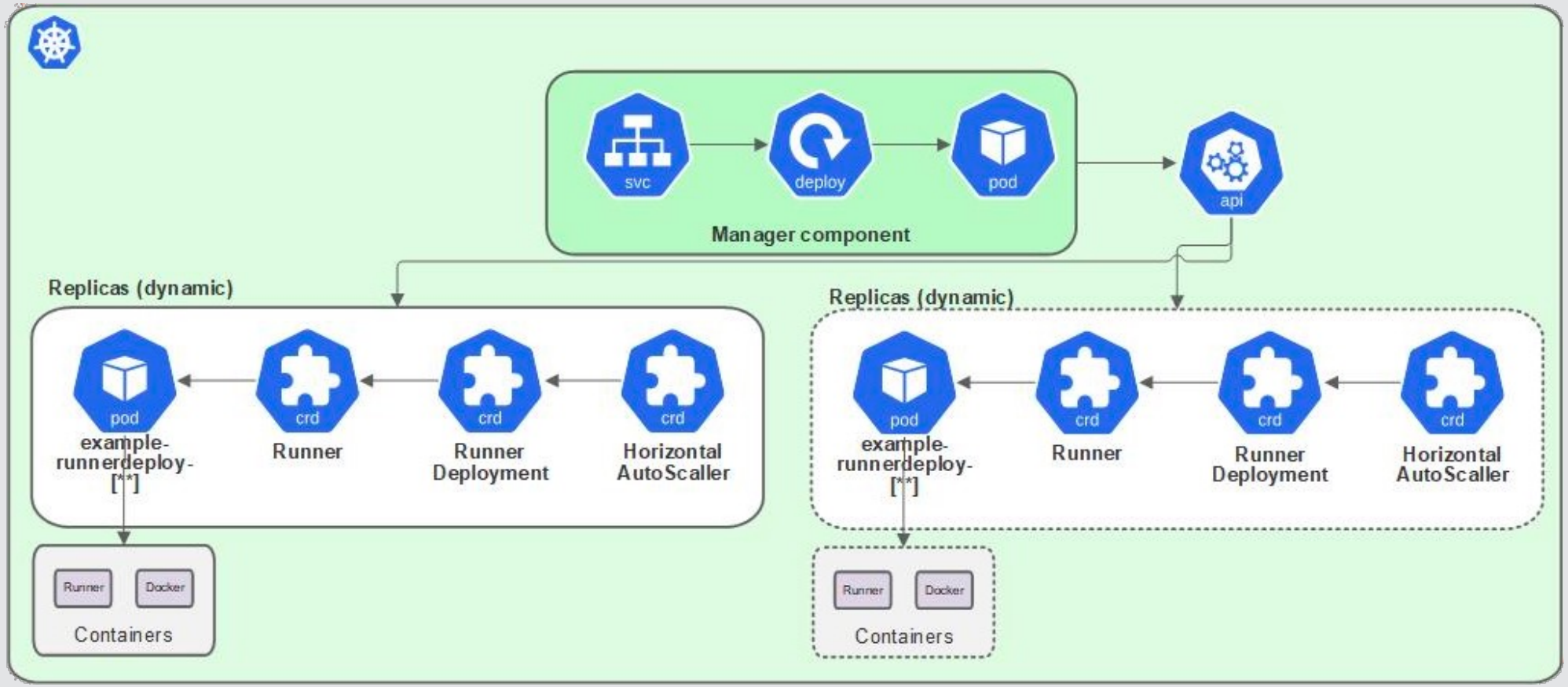


1 – Why self-hosted?

- **GitHub Enterprise Server** (self-hosted GitHub)
- **Custom hardware**
    - ARM processors
    - GPU compute
    - Testing hardware attached to a real machine
- **Custom software** beyond what's available or installable on the hosted runners
    - Red Hat Enterprise Linux
    - In-house Linux distributions
- Needing to run jobs in a specific environment, such as "gold load" type imaged machines
- Because you want to and ✨ **I'm not here to judge you** ✨

Actions-Runner-Controller

- Open-source project – https://github.com/actions/actions-runner-controller
- 🎉 Officially the **auto-scaling solution for self-hosted runners** for GitHub! 🎉
- Also ships some images for runners, most users build their own



Unique security challenges

Huge variety in software development and incumbent tools – hard to provide opinionated guidance.

Enables an anti-pattern where **containers can be used in very VM ways**

- Increases vulnerability area
- Increases “noise” from container scanners in your security tooling
- Caching for runtime speed can make for big container images

Combined with **economic incentives encourage poor security** posture

- Poorly scoped permissions or deployments
- Privileged pods (are easier than rewriting things to be containerized)
- Disabling SELinux, seccomp, AppArmor comes up way more often than I’d like
- `image:semver+date` or literally anything except `image:latest`
- Internal package/container registries that are terribly out of date

3 types of Actions

JavaScript

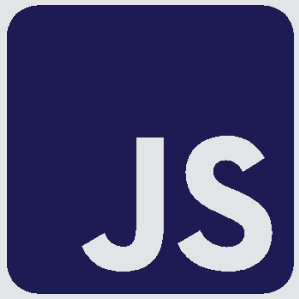
- Pure JavaScript
- Cannot rely on other binaries
- Node16 or Node12 (Node12 deprecating summer 2023)

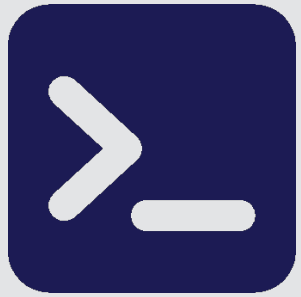
Composite

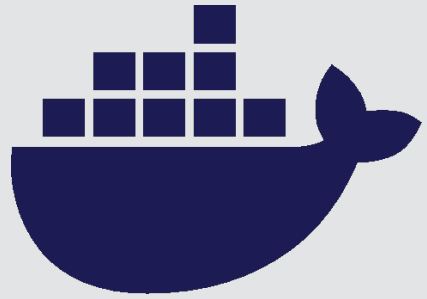
- Arbitrary languages supported
- In-line scripts
- Scripts or binaries stored on the runner or in a repo
- Assemble other Actions

Docker container

- Builds container on each run
- Executes container with inputs/outputs in `~/action.yml`
- Podman not a drop-in replacement for Docker!







3 types of security concerns

JavaScript

- Custom NPM registry config done on the pod
- Untrusted input (script injections) can do weird things

Composite

- In-line scripts
- Call binaries
- Do you allow users to use **sudo**?
- Can they install software?
- Mess with mounts?

Docker container

- What base image is being used?
- What's going on in that build?
- Docker-in-Docker requires privileged pods.

You're not just allowing privileged execution of random user input, right???

2 – Do you trust your neighbors?

Namespace isolation vs **Cluster** isolation

- Namespaces are great for **setting quotas and secrets** by group
- **RBAC, network and pod security policies**, etc., can be time-consuming to manage at scale
- **This is a tricky workload to isolate!**

Segregating privileged workloads is often the simplest solution.

Co-tenanting Kubernetes can be hard, sometimes it's **more economical to add another cluster** (or other hardware needed) to meet user demands.

--privileged :'(

Privileged pods are **very** common in actions-runner-controller implementations.

Why so common?

Docker-in-Docker is risky, but ...

Docker-in-Docker provides **the most "SaaS-like" experience**

- Users work with containers
- Docker Actions

There's some **alternatives and/or compensating controls** to consider

- Don't use Docker containers for GitHub Actions
- **Use VMs** for working with containers
- **Rootless (and sudo-less) Docker-in-Docker** pods (but still privileged!)
- **Firecracker** or Kata containers provide an OCI-compliant wrapper around VMs
- Use the **runner with K8s jobs** feature of actions-runner-controller
- Use **container builders** like Kaniko for users that only need to build/push containers

Rootless?

Rootless and **sudo-less** Docker-in-Docker

- Provide *some* coverage by removing the user's ability to run arbitrary shell commands as root.
- Still requires privileged pods to mount masks, modify SELinux/AppArmor, etc.
- Should only be used for jobs that need to run in a container.

```
runner@rootless-ubuntu-focal-tdkl8-k7455:/$ mount /dev/sda1 /tmp/test/
mount: only root can do that
runner@rootless-ubuntu-focal-tdkl8-k7455:/$ sudo mount /dev/sda1 /tmp/test
bash: sudo: command not found
runner@rootless-ubuntu-focal-tdkl8-k7455:/$ su -
Password:
su: Authentication failure
runner@rootless-ubuntu-focal-tdkl8-k7455:/$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:aa0cc8055b82dc2509bed2e19b275c8f463506616377219d9642221ab53cf9fe
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

nope.

still no.

no again.

this works still!

Example code here - https://github.com/actions/actions-runner-controller/blob/master/runner/actions-runner-dind-rootless.ubuntu-20.04.dockerfile

Just use  Firecracker

⚠️ **VMs are much more isolated than containers** ⚠️

Containers aren't VMs and **Kubernetes expects containers** – there are lots of papercuts on this route.

- Now you're (likely) running your own cluster over a managed service
- Volume mounts and other **things get weird**, vertical scaling is limited
- Firecracker-containerd improvements ship fast for Kubernetes!
- Fantastically documented project - https://github.com/firecracker-microvm/firecracker-containerd/tree/main/docs

Can still do silly things - it's just a little bit harder to do

- Customize seccomp to add privileges – like all of the privileges
- You can also disable seccomp entirely
- **Doing insecure things from a VM is still insecure!**

Runner with Kubernetes jobs

Runner with Kubernetes jobs **allows users to get out of needing privileged pods** in most use cases, but it's not a drop-in replacement for Docker-in-Docker.

Limitations

- Docker Actions don't work as-is
- All workloads on those runners will need a job container
- Container builds will need something like Kaniko

Learn more

- https://docs.github.com/en/actions/using-jobs/running-jobs-in-a-container
- https://github.com/actions/actions-runner-controller/blob/master/docs/deploying-alternative-runners.md#runner-with-k8s-jobs

Recommendations

- **Avoid privileged pods** altogether if possible. This may mean rewriting some of your existing workflows to be container-friendly.
- Segregate workloads that need privileged access.
- Use the **runner with Kubernetes jobs** in actions-runner-controller if possible.
- Rootless and sudo-less Docker-in-Docker is sometimes okay, but **it's still privileged!**
- Using VMs, even in the form of micro-vms with a container shim, is fine too – just know containers aren't VMs.

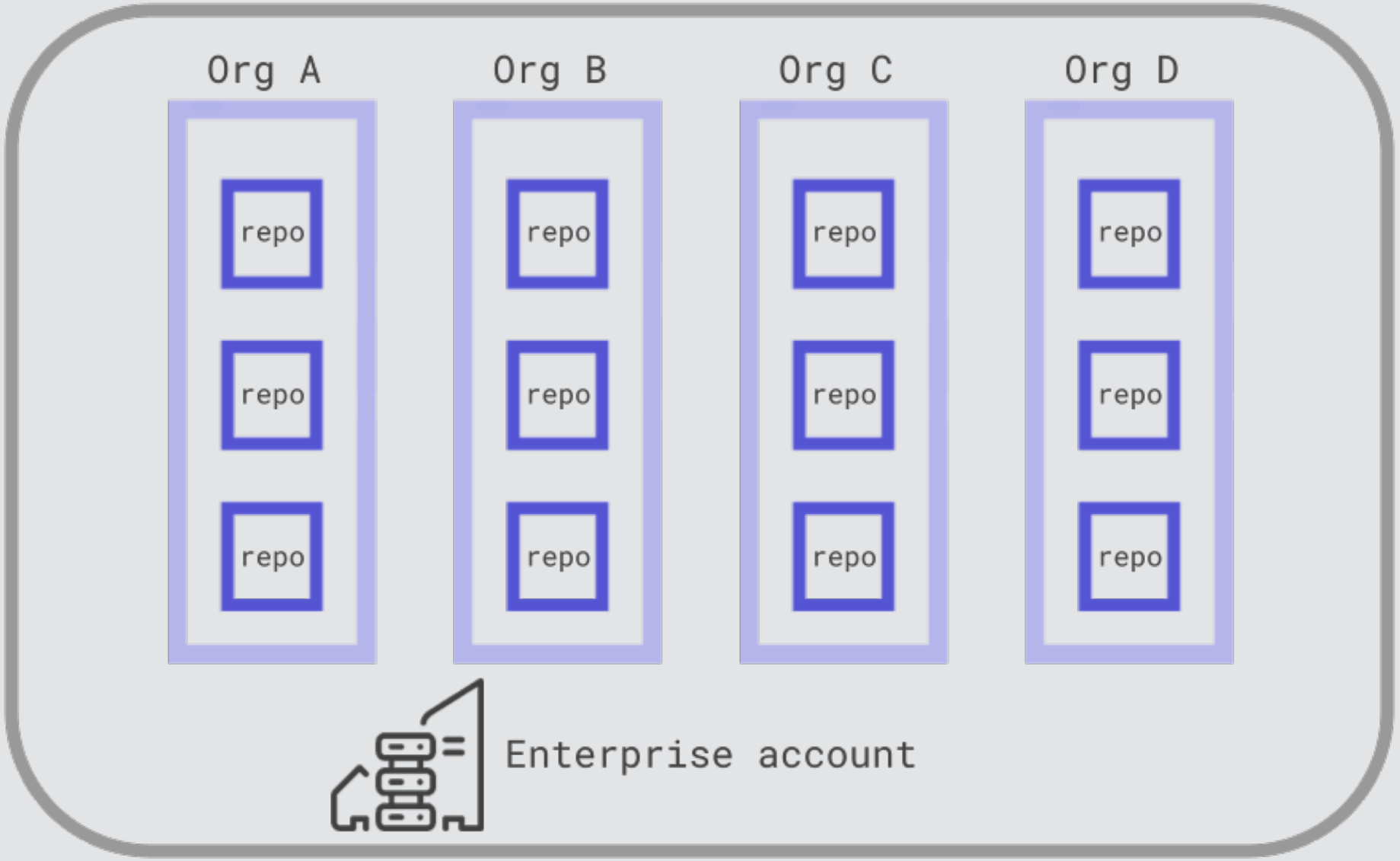
Multi-tenancy may come at the expense of security!

3 – Right-sizing your runners

Runners can exist at

- Enterprise
- Organization
- Repository

Every machine that can access GitHub on TCP/443 can be a runner.



Controller authorizations

Two authentication methods for actions-runner-controller

- **GitHub Apps** – granular control and higher API limits for repositories and organizations
- **Personal Access Tokens** (PATs) – similar more to a scoped password on a service account
    - Only auth method available to enterprise-wide deployments
    - Only grant it the scope needed (`manage_runners:enterprise`)

Apart from above, no difference in functionality.

⚠️ **Auth is stored as a secret in the namespace for the controller** (usually `actions-runner-system`).

It's passed to the pod to register it as a runner (this is **changing to JIT very soon**).

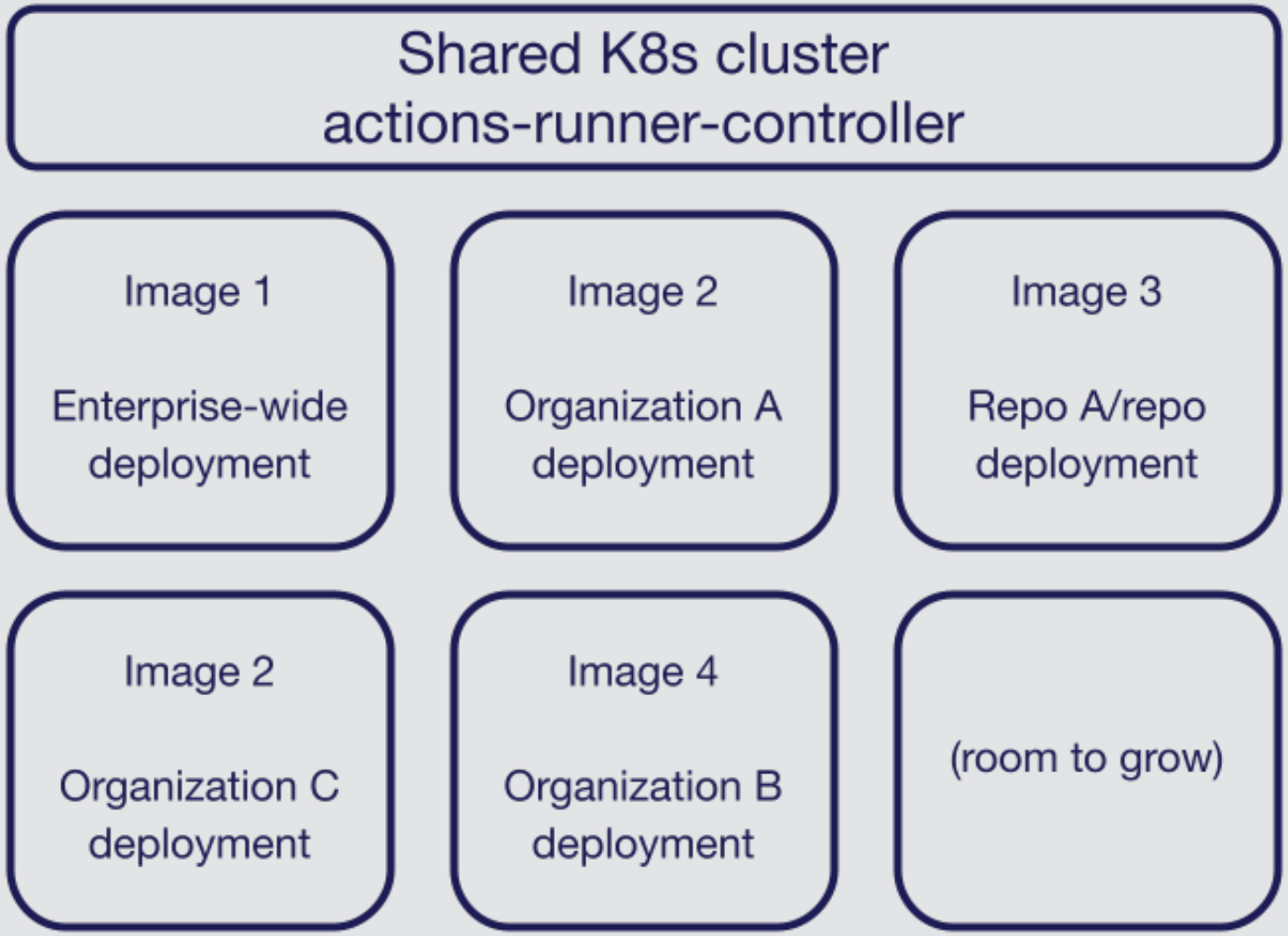
🤦‍♀️ **This **IS NOT** the token that GitHub Actions use at runtime!**

4 – Multi-tenant in practice

Cluster control
- Hardware management
- Ingress/egress
- Log forwarding

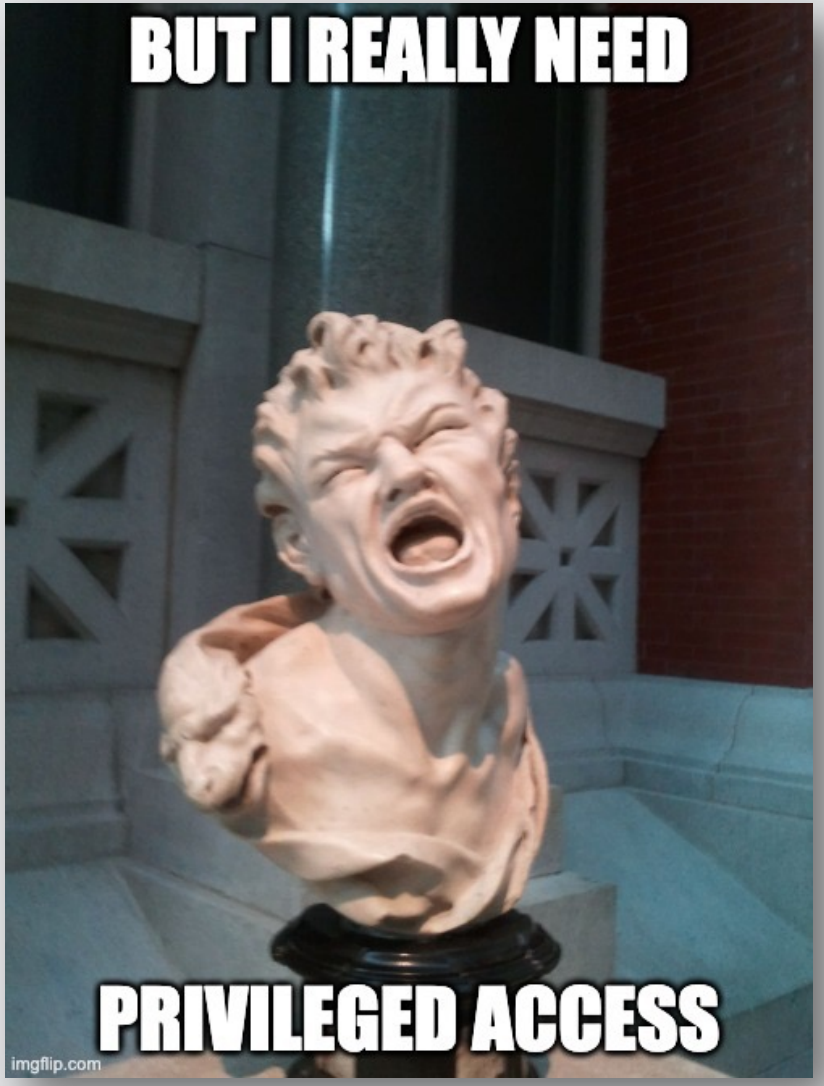
Namespaces set overall
- Resource quotas
- Secrets
- Policies

Deployments control
- Scope (who can use it)
- Image to use
- Vertical scaling
- Horizontal scaling
- Shared mounts

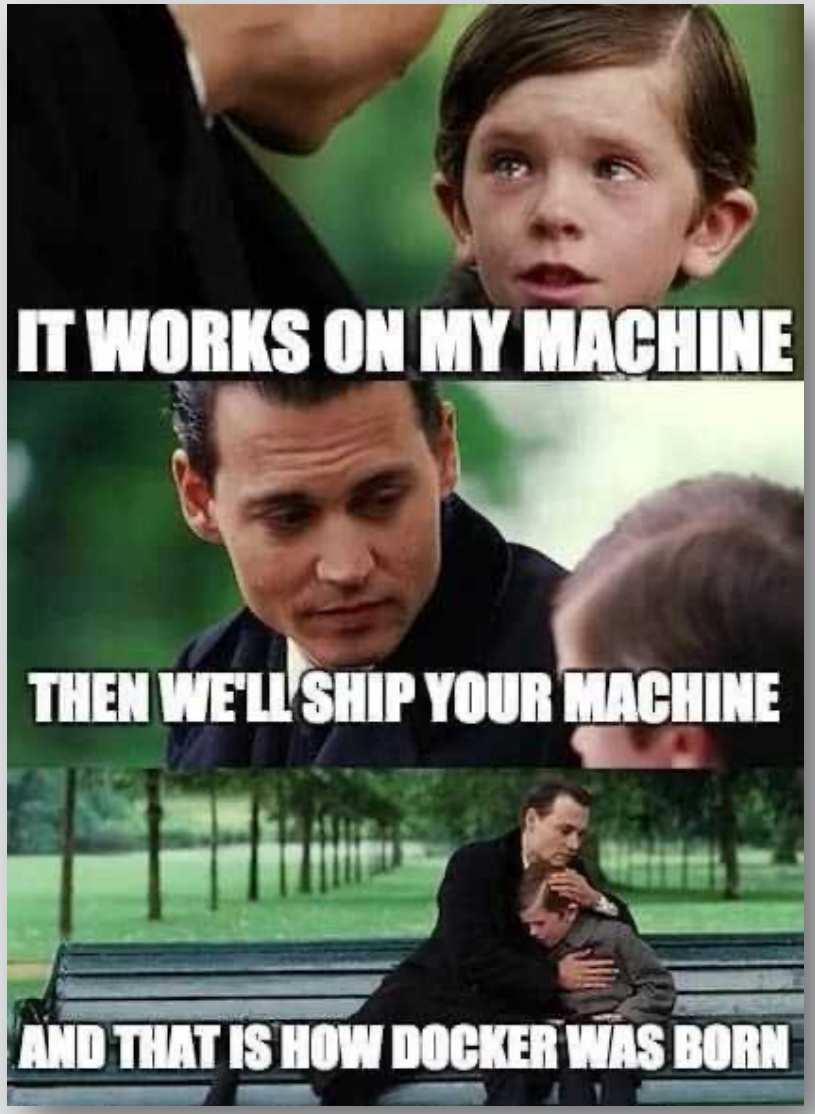


Recommendations

- **Default to safe** images for boring base usage at the available-to-everyone enterprise level.
- Allow (and govern!) **narrow deployments when needed**.
- Consider the maturity of teams with the number of discrete deployments vs number of admins.
    - Admin and security time is also very expensive!
    - More deployments of different runners is (slightly) more time spent building, scanning, testing, updating, etc.
    - **Economy of scale** versus **blast radius**
- Still encourages big pods … this isn't the worst thing ever.
- Friends don't let friends use `--privileged`



5 – Secure runner images



A typical runner image will usually have

- **Broad OS tag** as the image base (Ubuntu Jammy, UBI 8)
- **User account** setup (maybe adding user groups, sudo rights, etc)
- Runner agent and dependencies
- Copy in some shims and **other scripts**
    - Logging script
    - Entrypoint and/or startup script
    - Docker shim (for Docker-in-Docker)

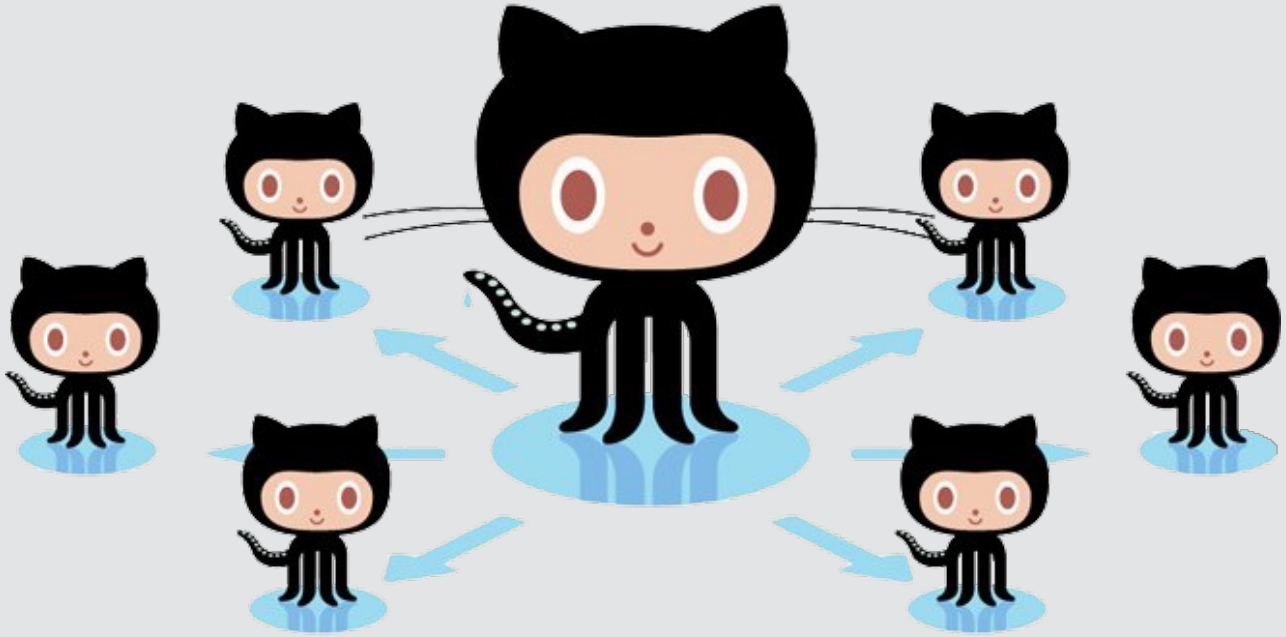
Sometimes included

- An init process (Yelp's `dumb-init` is popular, Red Hat `ubi-init` image)
- Dependencies users expect, installed with scripts / package manager
- Cached tools or dependencies

Examples to get started!

Actions-runner-controller includes 3 runner images, all based on Ubuntu, that you can use.

- Runner without Docker
- Runner with Docker-in-Docker
- Runner with rootless and sudo-less Docker-in-Docker



Tons more prior art on alternative runners - https://jonico.github.io/awesome-runners/

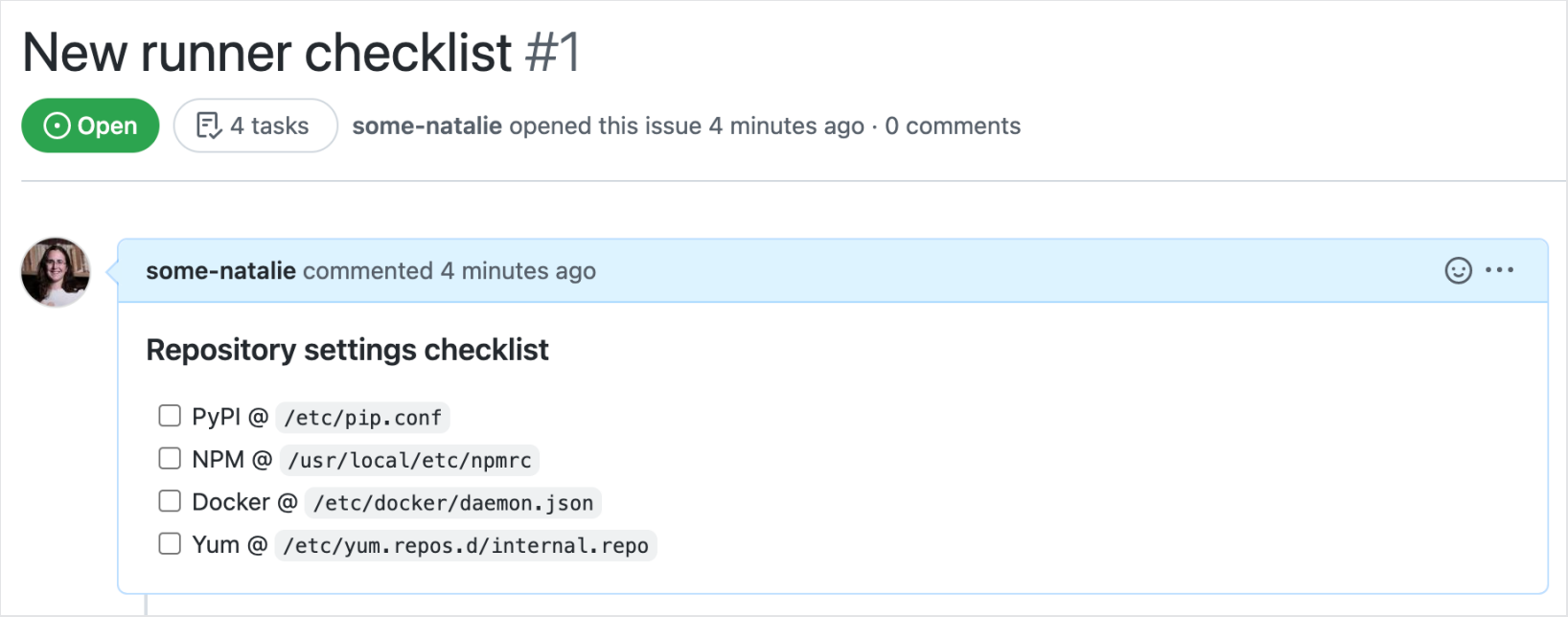
You may have forgotten

- Keep in-house repositories, base images reasonably **up to date**.
- Add **custom configurations** for package repositories to ALL the languages in use by the image.
- Self-signed **SSL certificates** or intermediaries?

  Add it everywhere!

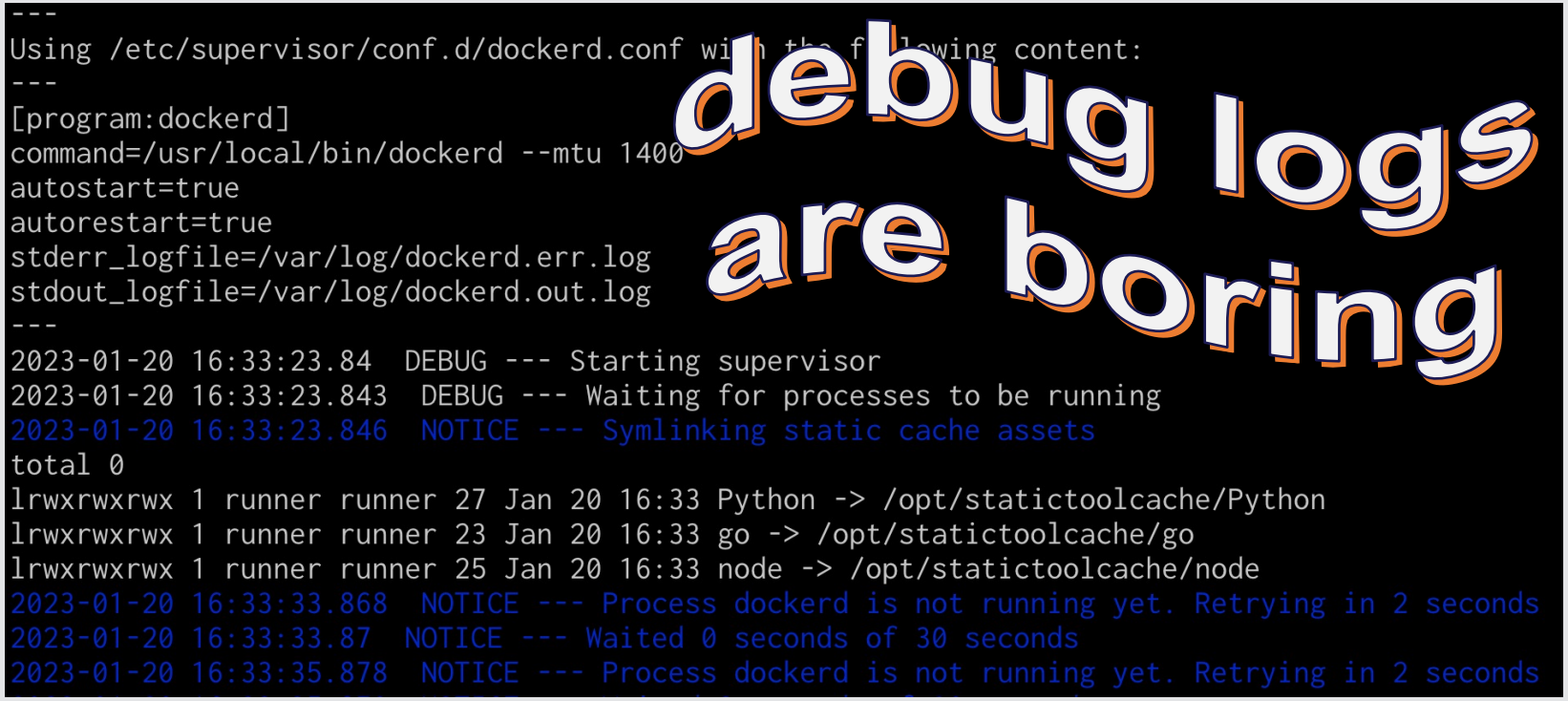
```
### Repository settings checklist

- [ ] PyPI @ `/etc/pip.conf`
- [ ] NPM @ `/usr/local/etc/npmrc`
- [ ] Docker @ `/etc/docker/daemon.json`
- [ ] Yum @ `/etc/yum.repos.d/internal.repo`
```



Logging is easy to overlook

- **What do you want to know?**
  - (easy) GitHub's audit logs = things that were done
  - (a little less easy) Actions run logs = console output from each run
  - Actions-runner-controller manager = 🤷‍♀️
  - Runner pods = 🤷‍♀️
- **Where are logs going?**
- **Sensible defaults for your runners**



Sharing (mounts) isn't caring!

Why would you ever ???

- Inner and outer Docker don't share a build cache
- `actions/setup-LANGUAGE` expects a bunch of stuff
- Dependencies in an ephemeral environment is wasteful
- I/O intensive builds can get super fast disks!!

Rate limiting is expensive.

Shared mounts aren't always version controlled!

Accidental data persistence at the intersection of **privileged pods** and arbitrary jobs **writing to disks.**

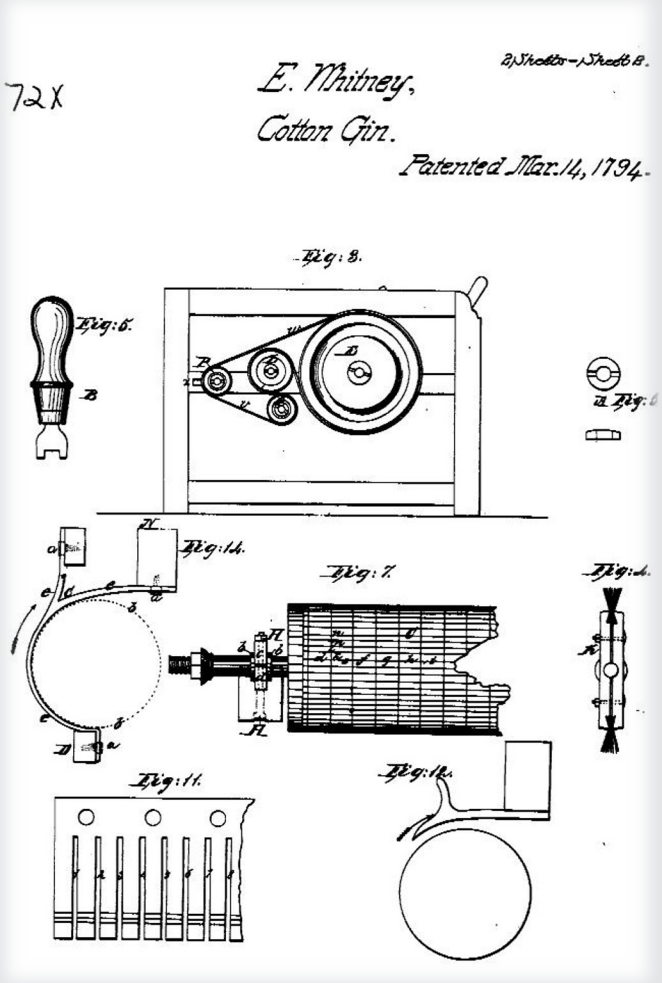
Building and deploying

Roughly, a pipeline to manage the runner images usually contains the following steps:

1. **Build it** – with what builds your containers
2. Scan it – container security (not VM/hardware) scans can fail the PR
3. Create and upload an SBOM – record what's in the thing
4. Sign it – record that this image really is what we say
5. **Tag it** – things get opinionated here
6. Test it – find what's broken before users
7. Ship it! – the squirrel has spoken

GitHub Actions is super modular.

Swap what's in place for scanning, signing, deployment, etc.



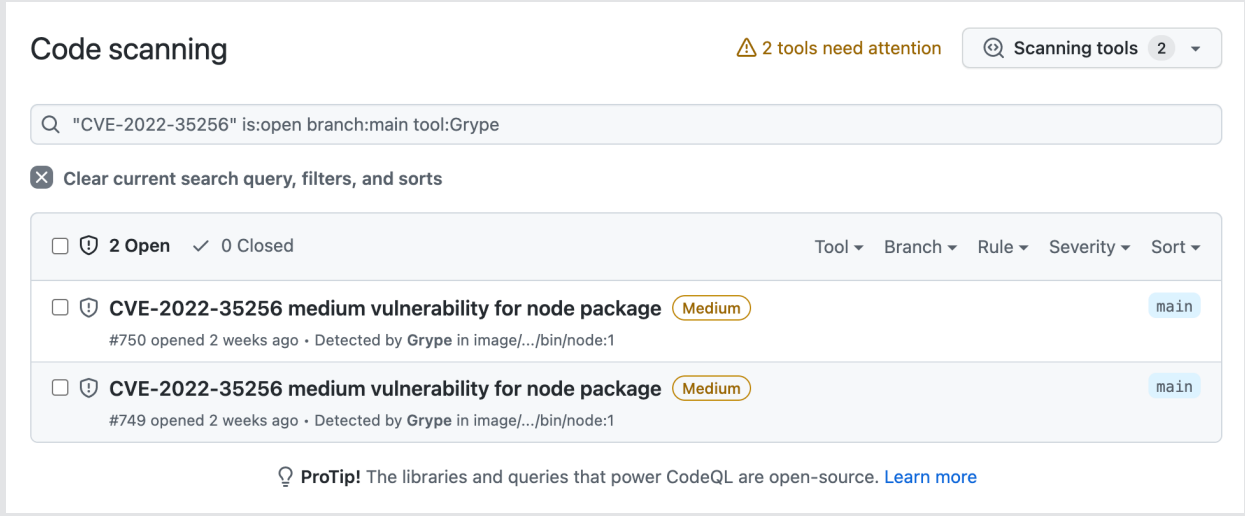
Sharing is caring!

Keeping the Dockerfile(s), deployment YAML, and other files in a shared **innersource** repo, then having Actions build/test/deploy itself **is remarkably powerful**.

- Git repo is the history of who requested/reviewed/approved what changes and when.
- Hey, can we add `<PACKAGE>` to all images? It's really helpful for everyone.
- My project needs `<FRAMEWORK>` and this other team already has an image.
- My build is failing on image `<name:tag>`. Here's a PR with tests to prevent this going forward.
- Show me all images vulnerable to `CVE-de-jour`. Right now.

Tagging by **semver+date** seems to work well.

- Bump semver for software changes.
- Bump date for routine rebuilds.
- Just say no to `image:latest`



6 – (in)conclusions

Balance granting permissions to users to

- Change their runner images
- New images / software
- Network exceptions

By scope – enterprise-wide, organization, or repository

If only there was a way to propose and review changes to code … 🤔

Your chances of "**shadow IT**" go up dramatically the longer a user request goes without evaluation/decision!

Questions!

CLOUDNATIVE SECURITYCON NORTH AMERICA 2023

Resources

GitHub docs

- Security hardening for GitHub Actions - https://docs.github.com/en/enterprise-cloud@latest/actions/security-guides/security-hardening-for-github-actions
- Autoscaling with self-hosted runners - https://docs.github.com/en/actions/hosting-your-own-runners/autoscaling-with-self-hosted-runners

Actions-runner-controller

- https://github.com/actions/actions-runner-controller
- Has lively issues and discussions sections

My website

- https://some-natalie.dev
- I work at a fun intersection of security, technology, and compliance – so I write about it! This presentation (slides, commentary, and more resources) will be there soon.