What are GitHub Actions?

- Custom workflows triggered on any GitHub event
- Workflows stored as `yml` files
- Live logs and visualized workflow execution
- Community powered workflows
- Linux, macOS, and Windows hosted runners
- Self hosted runners
- Built-in secret store
- Reusable workflows
- Parallelized job execution
Anatomy of a GitHub Workflow

WORKFLOW  TRIGGERS  JOBS  STEPS  ACTIONS
Anatomy of a GitHub Workflow
Quick Summary

• Triggers “trigger” workflows, e.g. a push to a branch
• Workflows contain one or more jobs, which contains one or more steps
• These steps can reference actions or execute commands
• The term “GitHub Actions” include all components, not just the Actions themselves
GitHub Marketplace

- Discover open-source Actions across multiple domains
- > 10,000 Actions (and counting)
- Verified creators
- Reference these Actions directly in your workflow
- Integrated into the GitHub editor
Reusable Workflows

• Workflows can be made reusable to avoid duplication and make it easier to maintain them
• Reusing workflows can help create new workflows more quickly and promote best practices
• A "caller" workflow uses a "called" workflow, and a single caller workflow can use multiple called workflows
• Reusing a workflow means using the entire called workflow as if it was part of the caller workflow
• If a workflow is reused from a different repository, actions in the called workflow run as if they were part of the caller workflow
GitHub Action Drawbacks

- Secret management is at the repository level, this means the same secret would have to be copied to each repository. This can be hard to manage without automation
- No native ARM support, would need to do some sort of virtualized build via qemu for ARM images
- Learning curve may be higher for developers not used to CI/CD
- Reusable workflows can become complicated and hard to manage for the developer community
- Less support from Linux Foundation
- Signing Git Tags may be difficult given the Sigul setup