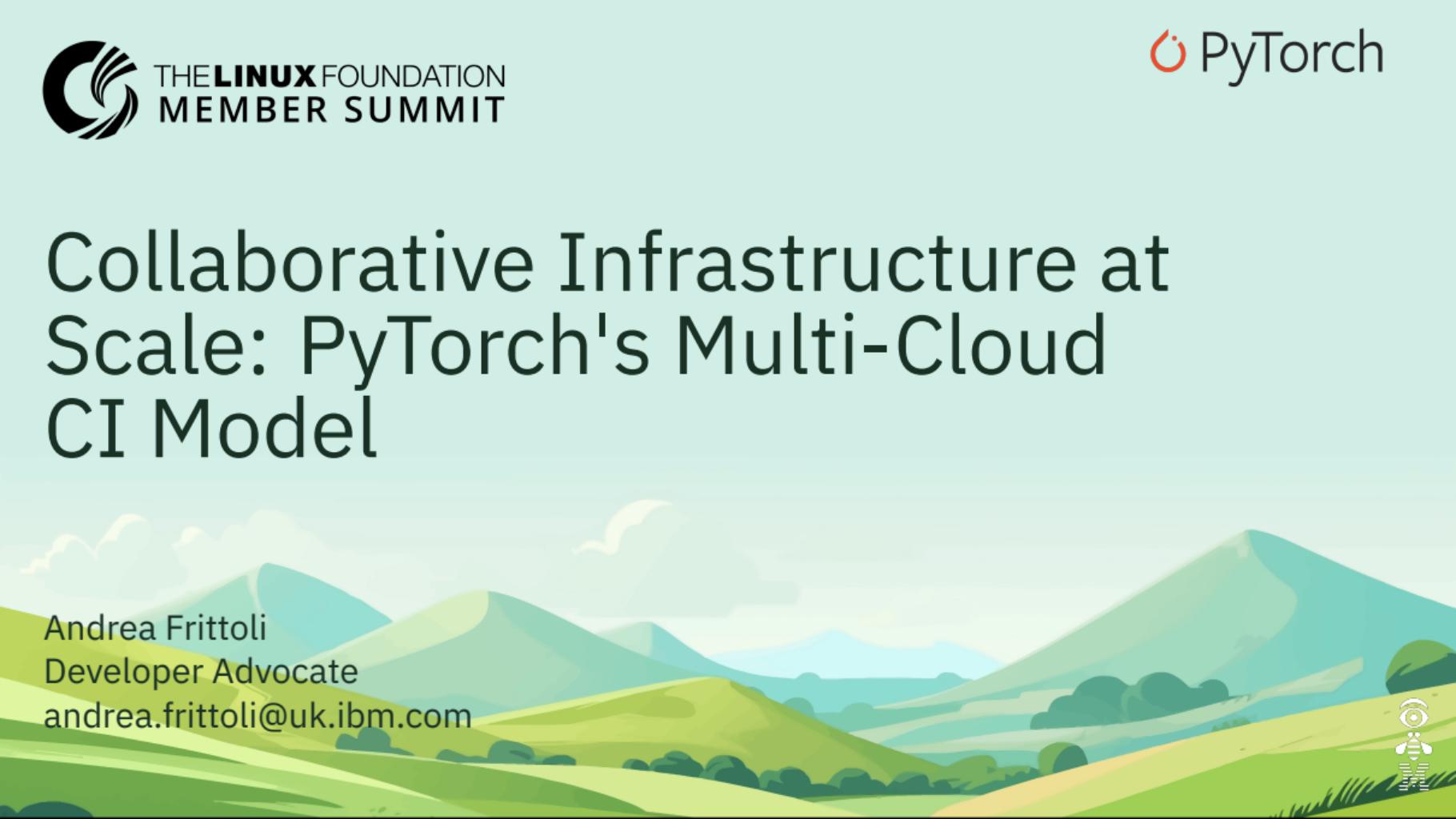


# Collaborative Infrastructure at Scale: PyTorch's Multi-Cloud CI Model



Andrea Frittoli  
Developer Advocate  
[andrea.frittoli@uk.ibm.com](mailto:andrea.frittoli@uk.ibm.com)



# Andrea Frittoli

⌚ afrittoli | 💬 andreafrittoli | X @blackchip76

- › Open Source Advocate @ IBM
- › Lives in Wales, enjoys the wind
- › Multi-Cloud CI Working Group Lead
- › Member of PyTorch Infra Working Group
- › Tekton, CDEvents maintainer





## Contents

The Infrastructure Challenge

PyTorch Foundation

Governance Model

Technical Architecture

Results & Next Steps

Q&A



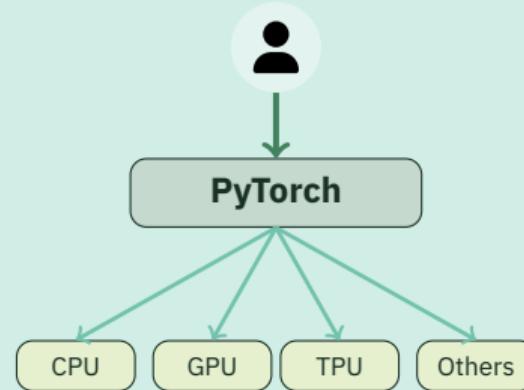
# The Infrastructure Challenge



# What is PyTorch?

## A Leading Open Source ML Framework:

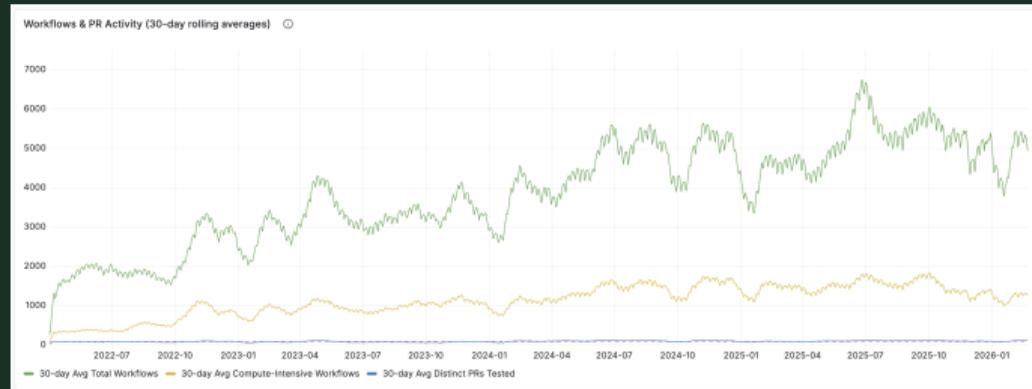
- › Researchers, data scientists, ML engineers
- › Research and production ML workloads
- › (Distributed) Training, LLM, RL, Inference
- › Python API, C++ Core (libtorch)
- › Eager and Graph Mode (Inductor)
- › Rapid growth and adoption across industry



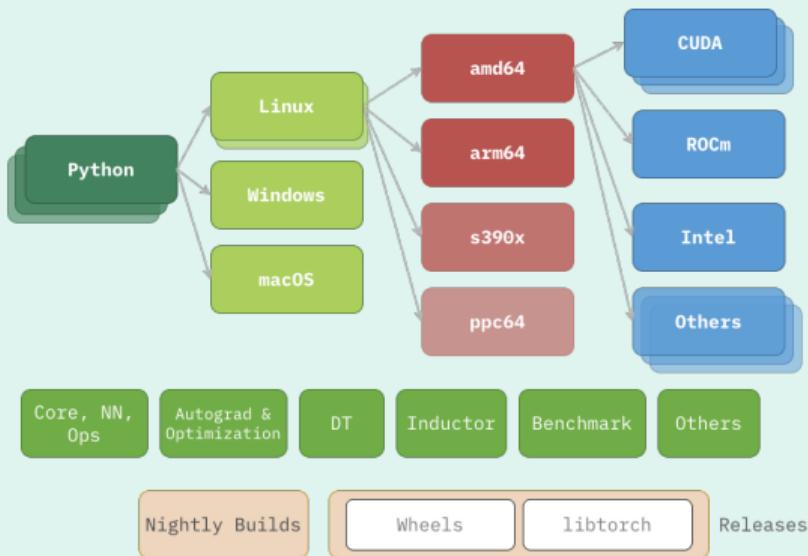
# CI/CD Infrastructure Today

- ~1M\$ monthly infrastructure costs
- ~1M hours of compute per month
- ~100 distinct PRs/day
- ~1.3k Compute-intensive Workflows/day
- Growing year over year

pytorch/pytorch repo only, numbers are estimated

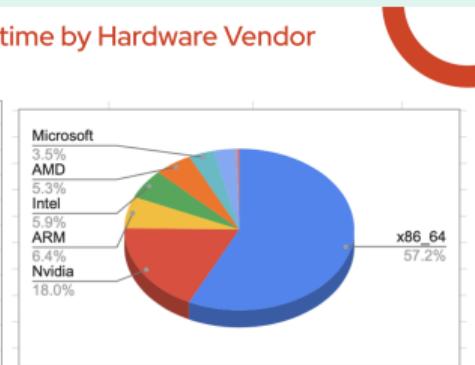


# An Expanding Test Matrix



Combined Hardware Runtime by Hardware Vendor  
September 2025

Combined Runtime By Hardware Vendor	
Vendor	Total
x86_64	834,085 hours
Nvidia	262,878 hours
ARM	93,000 hours
Intel	86,090 hours
AMD	77,517 hours
Microsoft	51,515 hours
Apple	46,812 hours
IBM	6,245 hours
N/A	0 hours
1,458,142 hours	



\* Data from PT Foundation AWS Account + PyTorch HUD. This does not include self-hosted runner costs by member provided runners.



# The Challenges

## Community Challenges:

- › Maintain high-quality end-user experience
- › Preserve contributor workflow
- › Provide clear path for vendor engagement

## Scale Challenges:

- › **Engineering:** Managing a Diverse Fleet, Access to Platforms
- › **Financial:** Reconcile Community Wishes with Budget Constraints

## Security Challenges:

- › Trust and Consistency in a diverse environment
- › Central Administration
- › Trusted Builds

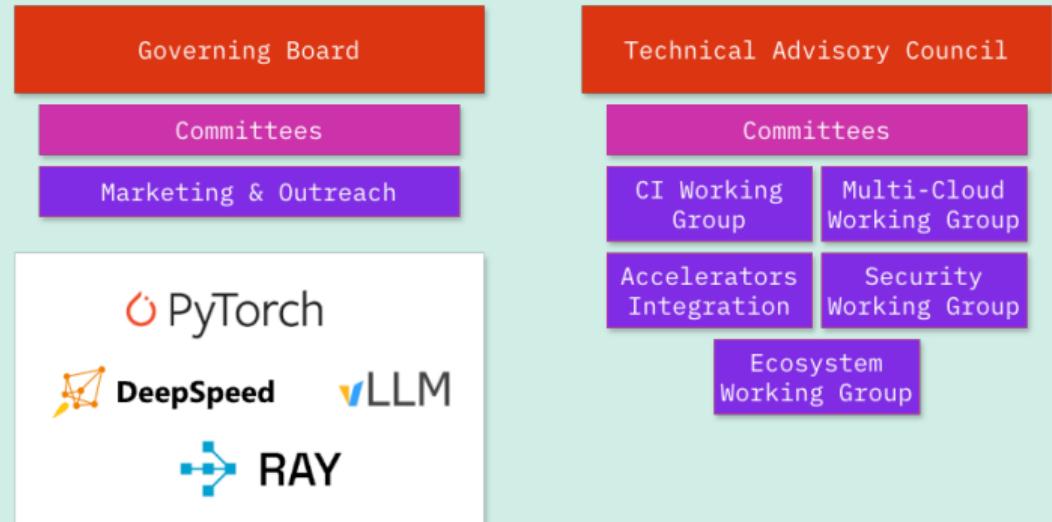


# PyTorch Foundation & Working Groups



# The PyTorch Foundation

- › Part of the Linux Foundation
- › Neutral home for PyTorch project
- › Hosts multiple projects
- › Thought Leadership (GB)
- › Technical Leadership (TAC)
- › Marketing & Outreach



# Working Groups

## Multi-Cloud CI

Cloud Agnostic Infrastructure:

- CI/CD jobs and Supporting Infra
- Metrics and Monitoring
- Fleet Management

Vendor-managed Runner Pools

## CI Infra

Responsible for the CI/CD infra:

- Develop and maintain tools
- Operate the AWS/GitHub fleet
- Execute Releases

## Accelerator Integration

Software Integration:

- Developer Guide
- Framework Improvement

Reference Test Framework

CI Event Relay Infrastructure

## Security

- PyTorch Security Triage
- Tools and reports
- CI/CD Security



# Governance Model



Photo by Khampha Phimmachak, CC0



# Personas: A Balancing Act

## End Users

- Stable software
- Available on my dev platform
- Available on my prod platform
- Following the industry at speed

## Platform Vendors

- PyTorch for my platform
- Demonstrate platform compatibility
- Low bar to contribution of infrastructure
- Clear guidance and processes

## Project Maintainers

- Project Success
- Stay relevant for end-users
- High quality secure builds
- Manageable CI/CD System (scale)
- Smooth CI/CD experience for contributors

## PyTorch Foundation

- Ensure Vendor Neutrality
- Vendor participation encouraged
- Fair representation
- Financial sustainability



# Integration Models

## Three Ways to Contribute Infrastructure:

- › **Public Cloud Credits:** Financial contribution for shared infrastructure
- › **Vendor-Managed Runner Pools:** Tightly integrated with GitHub Actions
- › **Vendor-Managed Test Infrastructure:** Loosely integrated with GitHub

	Community	Scale	Security
Advantages	Flexible models accommodate different vendor capabilities	Outsourcing engineering with vendor-managed setups	More loosely coupled systems cannot compromise core CI system
Challenges	Hide the underlying infra complexity to contributors	Consistent tech stack required for public cloud credits. Consistent metrics and monitoring across the board.	Ensure security best practices on vendor managed infrastructure



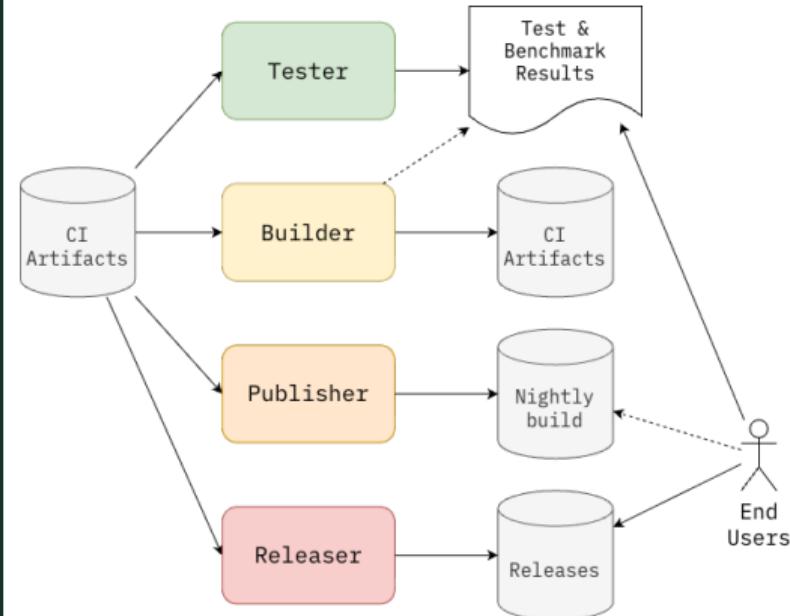
# Roles & Requirements

## Least privilege access to cloud resources:

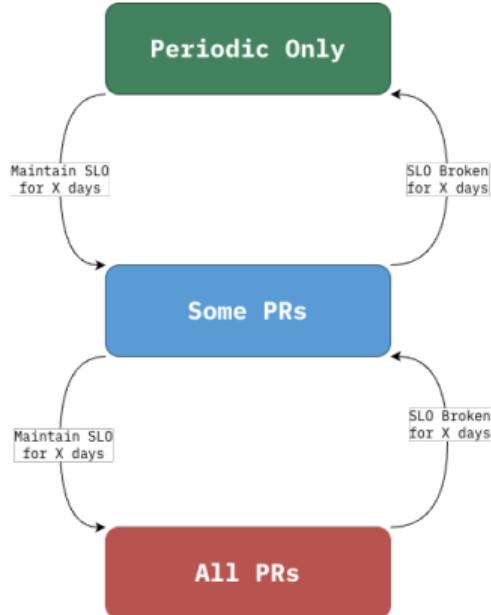
- › Jobs specify nodes access level required
- › Runners are assigned a role
- › Roles define the level of access to resources

## Requirements for Runners:

- › Configuration
- › Provisioning
- › Monitoring
- › Security



# Triggers & SLOs



Trigger levels for CI/CD jobs:

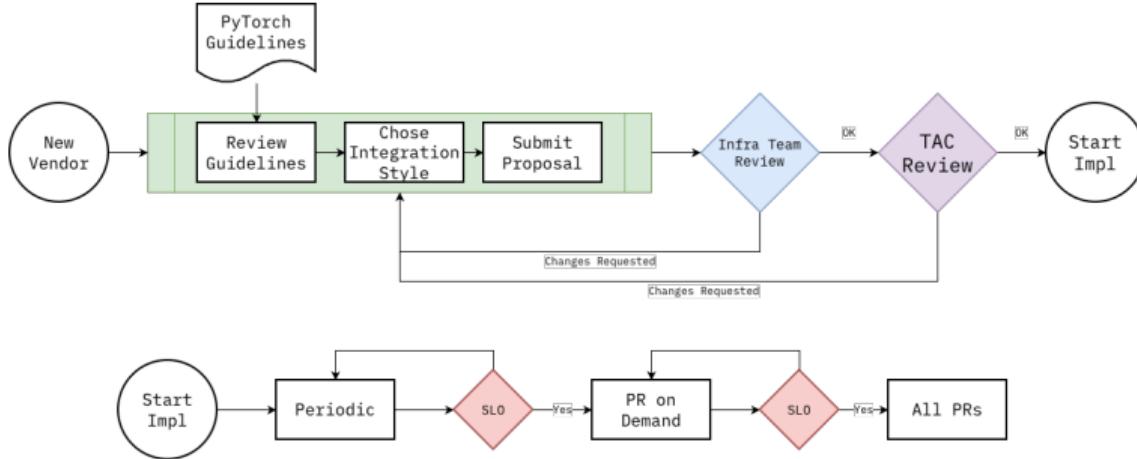
- On-demand only
- Periodic nightly
- Periodic multi-times per day
- Specific PRs only
- All PRs

Service Level Objectives:

- Availability %
- Reliability (Num of infra failures)
- Average Job Queue Time (p95/3 months)
- Engineering Commitment (on-call, slack, meetings)



# Contribution Process



- › Working Group provides guidance and recommendations
- › Infra team verify checks for compliance
- › TAC Community for final approval
- › Transparent process with clear requirements
- › Monitoring for SLOs



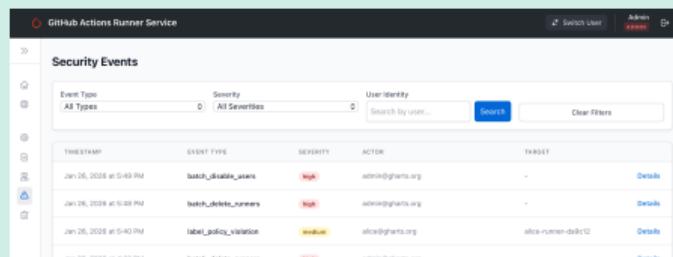
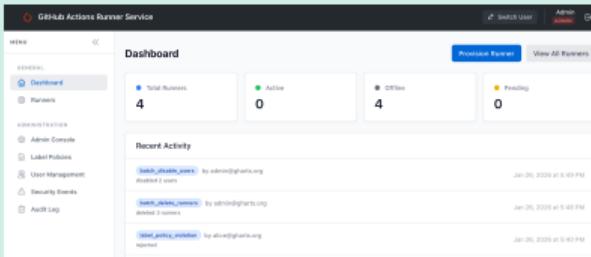
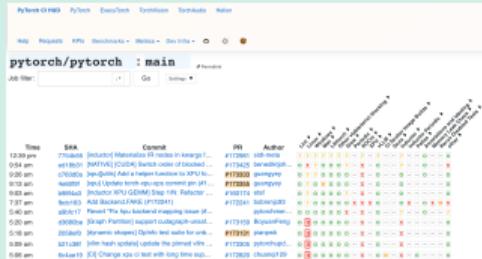
# Central Administration & Visibility

## Maintaining Manageability at Scale:

- › Common software stack across clouds  
Reusable by vendors too
  - › Self-service Tools with Central Admin Override
  - › Vendor playbooks, reusable IaaC

## Visibility & Control:

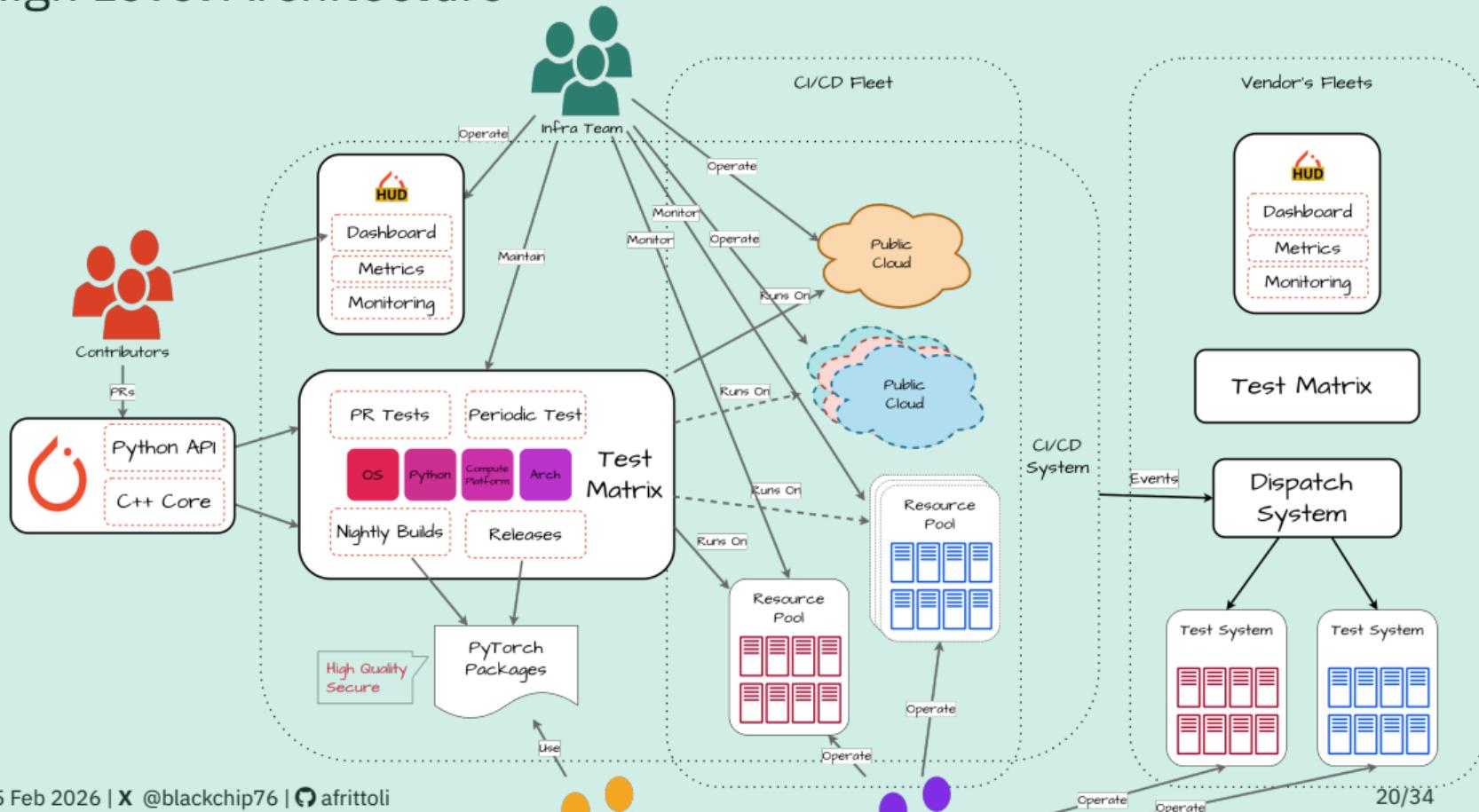
- Public Dashboards: Transparency for community and vendors
  - Cost Tracking: Attribution per vendor and workload type
  - Performance Metrics: Build times, success rates, SLA compliance



# Technical Architecture



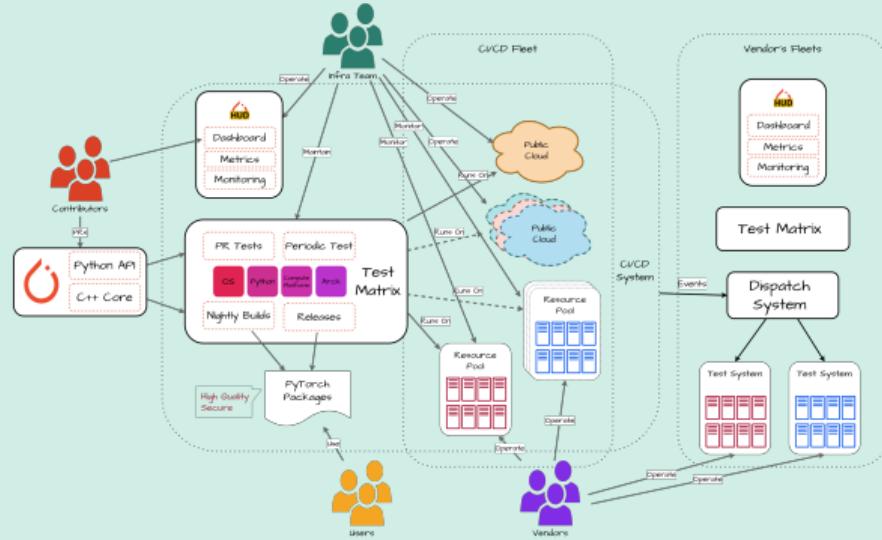
# High Level Architecture



# Public Cloud Infrastructure

## Current AWS Setup:

- › Primary infrastructure hosted on AWS
- › Two accounts, funded by AWS Credits and Meta
- › Self-hosted GitHub Actions runners
- › Managed by PyTorch CI/CD team
- › Autoscaling based on workload demand



## Multi-Cloud & Infra Working Group Initiatives:

- › **Portable CI Jobs:** Run in a container, remove cloud-specific assumptions
- › **Reusable Autoscaler:** Developing portable autoscale based on ARC
- › **Infrastructure as Code:** Reusable



# Vendor-Managed GitHub Runners

## GitHub Actions Runner Token Service (GHARTS):

- Secure, self-service API to register self-hosted runners
- No long-lived credentials, on demand short-lived tokens
- Enforcement of runner metadata (label, group) and quotas
- Central administration, bulk operations
- Audit logging for all runner operations
- MVP being deployed, IBM to trial service

The screenshot shows the Tekton Hub interface. At the top, there's a search bar with the word "build" and a "Login" button. Below the header, a banner says "Welcome to Tekton Hub" and "Discover, search and share reusable Tasks and Pipelines". On the left, there's a sidebar with filters for "Sort By", "Kind" (Task), "Catalog" (Tekton), and "Category" (Automation, Build Tools, CLI, Cloud, Deploy, Editor, Git, Image Build, Language, Messaging, Monitoring, Notification, Others, Security, Storage). The main area displays a grid of tasks. Each task card includes a user icon, a star rating, the task name, its version, a brief description, the last update time, and a blue "View Task" button.

Task Name	Version	Description	Last Updated	Action
buildid	v0.1	Given a base version, this task generates a unique build id by appending the base-version to the current timestamp.	Updated 3 months ago	View Task
buildah	v0.2	Buildah task builds source into a container image and then pushes it to a container registry. Buildah Task builds source into a container image using Project Atomic's Buildah build tool.	Updated 3 months ago	View Task
buildkit	v0.1	This Task builds source into a container image using Moby BuildKit.	Updated 3 months ago	View Task
buildpacks	v0.2	The Buildpacks task builds source into a container image and pushes it to a container registry.	Updated 3 months ago	View Task
buildpacks-phases	v0.1	The Buildpacks-Phases task builds source into a container image and pushes it to a container registry.	Updated 3 months ago	View Task
golang build	v0.1	This Task is Golang task to build Go projects.	Updated 3 months ago	View Task

A JSON object representing a Task definition:

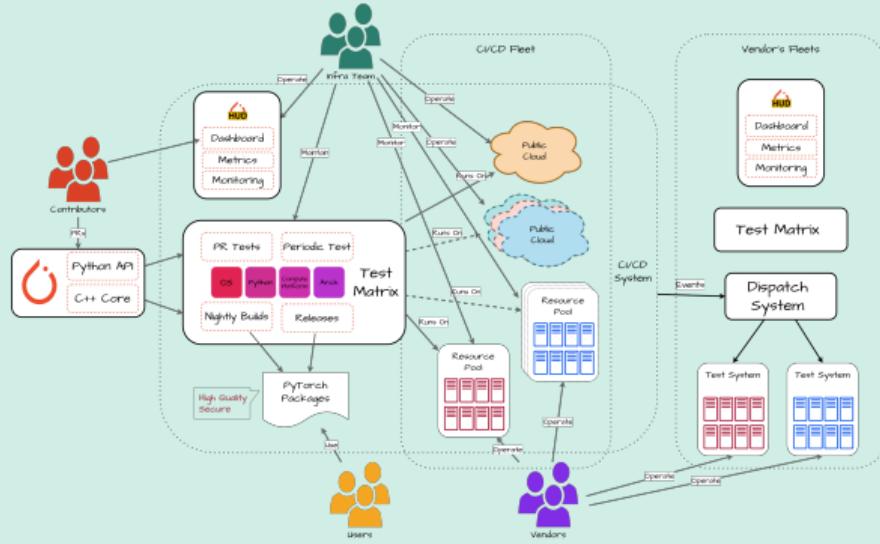
```
{
  "id": 33,
  "name": "git-cli",
  "catalog": {
    "id": 1,
    "name": "tekton",
    "type": "community"
  },
  "kind": "Task",
  "latestVersion": {
    "id": 1,
    "name": "git-cli",
    "version": "v0.1"
  }
}
```



# GHARTS Architecture

## Workflow:

- Vendor authenticates using existing credentials (LFID)
- Vendor requests a JIT config from GHARTS using the JWT
- GHARTS verifies AuthN and AuthZ for vendor
- GHARTS requests JIT config from GitHub
- Vendor uses JIT config to provision a runner
- JIT lasts 1h, and can only be used Once



# Vendor-Managed Test Systems

## **Accelerator Integration Working Group Initiative:**

- › Loosely coupled with GitHub Actions
- › Vendors provide test infrastructure and environments
- › CI triggers tests via webhooks or API calls
- › Results reported back to GitHub

## **Use Cases:**

- › Specialized hardware testing (TPUs, custom accelerators)
- › Long-running performance benchmarks
- › Platform-specific integration tests
- › Vendor-specific optimization validation



# Test System Visibility

## Challenges:

- › Loosely coupled systems harder to monitor
- › Results come from external systems
- › Need consistent reporting format
- › Debugging failures more complex

## Ongoing Work:

- › Standardized result reporting via GitHub workflow\_run events
- › Exploring alternatives: OpenTelemetry for metrics, CDEvents for event streaming
- › Unified dashboard for cross-vendor visibility
- › Real-time event relay for debugging



# Results & Lessons Learned



# Platform Coverage Expansion

## Achievements:

- › **Public Cloud Credits:** AWS, Meta funding
- › **Runner Pools:** IBM (via GHARTS), additional vendors onboarding
- › **Vendor Systems:** AMD, Intel, NVIDIA, ARM for specialized accelerator testing
- › Significant increase in platform coverage across CPU architectures and accelerators
- › Reduced time-to-market for new platform support
- › Improved test reliability through dedicated infrastructure



# Vendor Onboarding Experience

## Positive Feedback:

- › Clear expectations and requirements documentation
- › Reasonable onboarding timeline with good support
- › Responsive technical support from maintainers
- › Fair and transparent governance model

## Ongoing Initiatives:

- › IBM trialing GHARTS for runner pool integration
- › Red Hat collaborating on vendor system visibility improvements
- › Vendors actively sharing experiences and best practices



# Key Lessons Learned

## **Principles for Collaborative Infrastructure:**

- › **Fair Governance:** Protect everyone's interests, vendors can't veto but have voice
- › **Engage Vendors Early:** Let them help build solutions, not just consume them
- › **Empower Maintainers:** Give infra team authority and tools to manage at scale
- › **Security First:** Build security in from day one, not as afterthought
- › **Flexible Models:** Multiple contribution paths reduce friction and attrition

## **Challenges:**

- › Balancing vendor needs with project sustainability
- › Standardization across diverse platforms and requirements



# Applicability to Other LF Projects

## Shared Challenges:

- › Accepting vendor infrastructure
- › Maintaining project neutrality
- › Ensuring transparency
- › Achieving financial sustainability

## This Model Can Help:

- › Proven governance framework
- › Technical architecture patterns
- › Onboarding playbook



# Future Directions

## Ongoing Activities:

- › Complete GHARTS rollout with IBM and additional vendors
- › Enhance monitoring and observability across all integration models
- › Expand portable CI/CD tooling for multi-cloud deployments
- › Standardize event streaming with OpenTelemetry/CDEvents

## Long-term Goals:

- › Share governance and technical patterns with other LF projects
- › Build community of practice around collaborative infrastructure
- › Continuously refine based on vendor and maintainer feedback



# Key Takeaways

- › **Sustainable Scaling:** Distribute infrastructure costs while maintaining project control
- › **Governance Matters:** Fair rules enable vendor participation without compromising autonomy
- › **Security & Transparency:** Non-negotiable foundations that build trust
- › **Flexible Models:** Multiple contribution paths reduce friction and increase adoption
- › **Reusable Pattern:** This model can help other Linux Foundation projects





# Questions?



# References & Contact

## Resources:

- › PyTorch Foundation: <https://pytorch.org/foundation>
- › PyTorch CI/CD: <https://github.com/pytorch/pytorch/wiki/CI>
- › GitHub: <https://github.com/pytorch/pytorch>
- › Multi-Cloud Infrastructure WG: PyTorch Slack #infra-wg

## Contact:

- › Andrea Frittoli
- › [andrea.frittoli@uk.ibm.com](mailto:andrea.frittoli@uk.ibm.com)
- ›  @blackchip76 |  afrittoli

