THE LINUX FOUNDATION MEMBER SUMMIT

PyTorch

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

Andrea Frittoli
Developer Advocate
andrea.frittoli@uk.ibm.com

Andrea Frittoli

afrittoli | andreafrittoli | @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

Photo by Calum Lewis, CC0

Contents

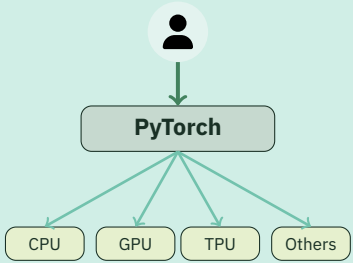
The Infrastructure Challenge

Photo by Denys Nevozhai, CC0

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



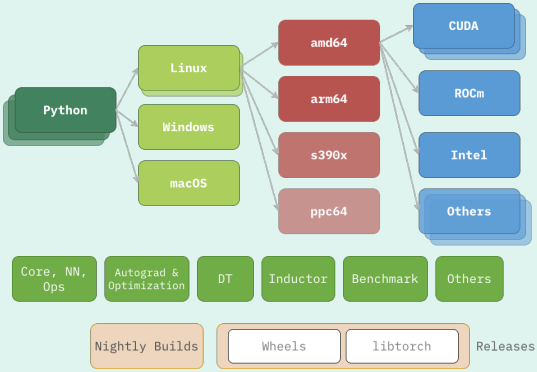
CI/CD Infrastructure Today

- ~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

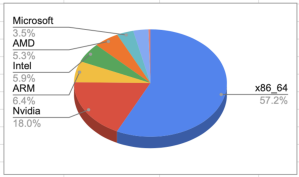
An Expanding Test Matrix



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

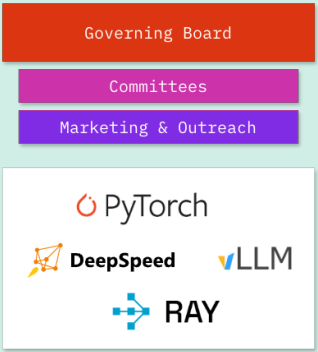
Security Challenges:

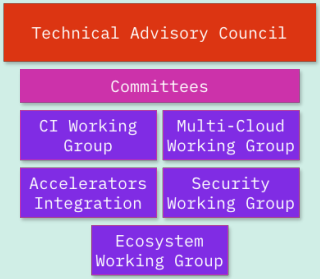
- Trust and Consistency in a diverse environment
- Central Administration
- Trusted Builds

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

Accelerator Integration

Software Integration:

- Developer Guide
- Framework Improvement

Reference Test Framework

CI Event Relay Infrastructure

CI Infra

Responsible for the CI/CD infra:

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

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:

- CI/CD Jobs specify nodes access level required
- CI/CD Runners are assigned a role
- Roles define the level of access to resources
- Roles identified: tester, builder, publisher, releaser

Requirements for Runners:

- Configuration
- Provisioning
- Monitoring
- Security
- All PRs

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
- Monitoring for SLOs
- Transparent process with clear requirements

Central Administration & Visibility

Maintaining Manageability at Scale:

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

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

Photo by Nadi Whatisdelirium, CC0

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 terraform modules

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

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:

- a
- b

Results & Lessons Learned

Platform Coverage Expansion

Achievements:

- X% increase in platform coverage
- New accelerator support (specific examples)
- Reduced time-to-market for new platforms
- Improved test reliability

Vendor Onboarding Experience

- Clear expectations and requirements
- Reasonable onboarding timeline
- Good technical support from maintainers
- Fair governance model
- Challenges: [specific examples]

Key Lessons Learned

What Worked:

- Clear governance from day one
- Security-first approach
- Flexible contribution models
- Transparent communication

What Was Hard:

- Balancing vendor needs with project needs
- Standardization across diverse platforms

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

What's Next:

- Expand to more vendors
- Improve automation and tooling
- Share model with other projects
- Refine governance based on experience
- Build community of practice

Key Takeaways

- Infrastructure costs can be distributed sustainably
- Vendor contributions work with right governance
- Project autonomy and vendor participation can coexist
- Security and transparency are non-negotiable
- Model is applicable to other LF projects

Questions?

Brecon Beacons, Wales, Photo by Carl Jorgensen, CC0

References & Contact

Resources:

- PyTorch Multi-Cloud CI Working Group: [URL]
- Documentation: [URL]
- GitHub: [URL]

Contact:

- Andrea Frittoli
- andrea.frittoli@uk.ibm.com
- @blackchip76 | afrittoli