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

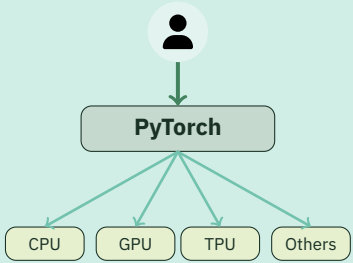
The Infrastructure Challenge

leonardo.ai, 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



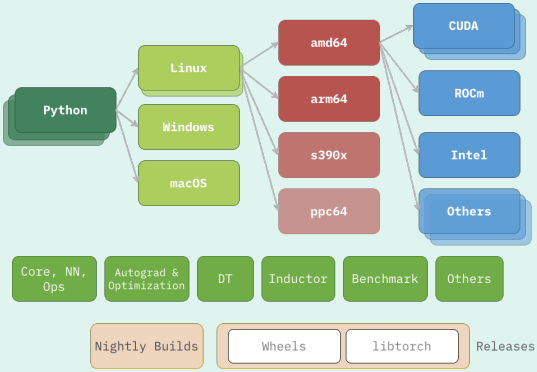
CI/CD Infrastructure Today

- ~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

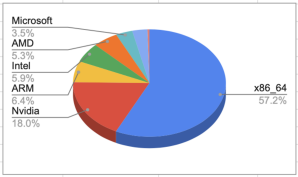
An Expanding Test Matrix



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

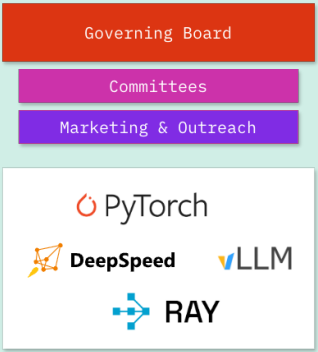
Security Challenges:

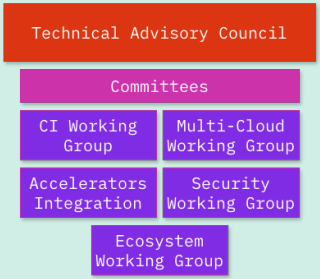
- Trust and integrity in a diverse environment
- Central Administration
- Trusted Builds

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

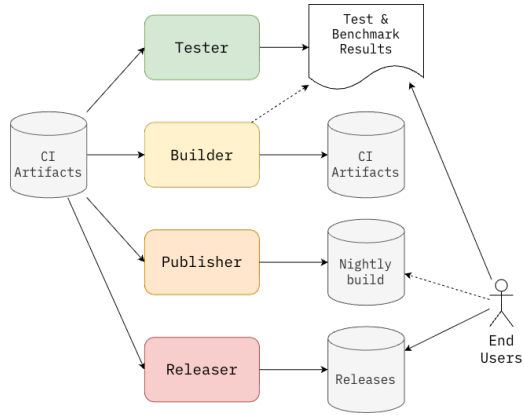
Roles & Requirements

Least privilege access to cloud resources:

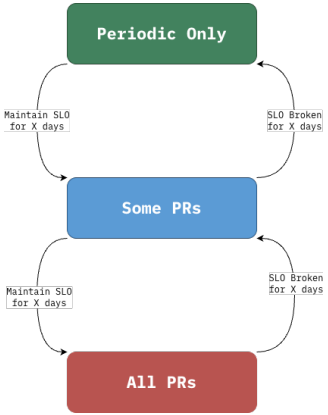
- Jobs specify nodes access level required
- Runners are assigned a role
- Roles define the level of access to resources

Requirements for Runners:

- Configuration
- Provisioning
- Monitoring
- Security



Triggers & SLOs



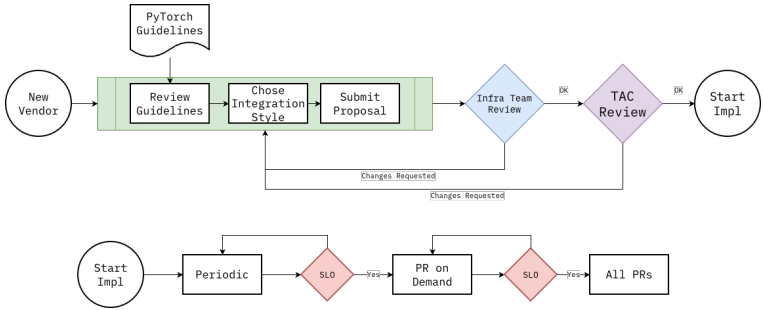
Trigger levels for CI/CD jobs:

- 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

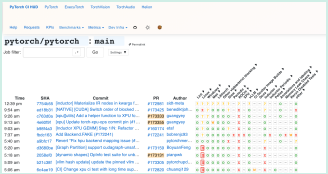
Central Administration & Visibility

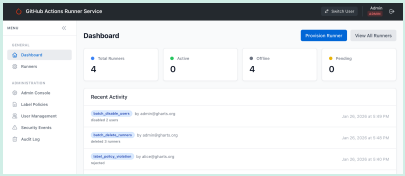
Maintaining Manageability at Scale:

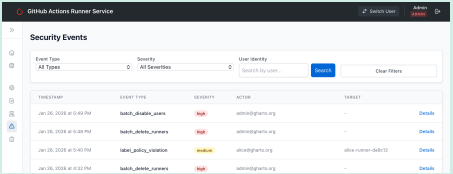
- Common software stack across clouds
  Reusable by vendors too
- Self-service Tools with Central Admin Override
- Vendor playbooks, reusable IaaC

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



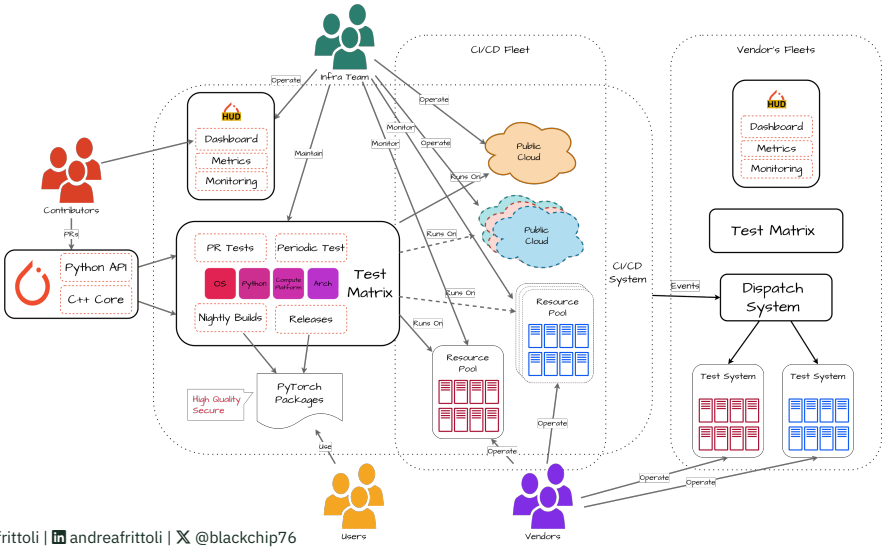




Technical Architecture

Photo by Nadi Whatisdelirium, CC0

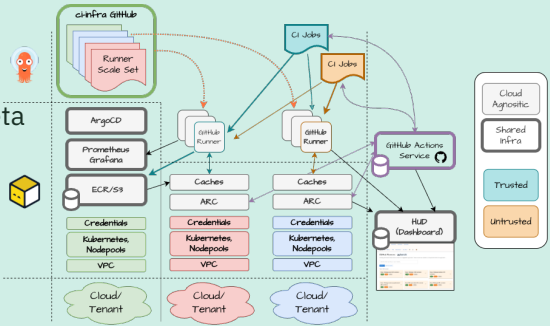
High Level Architecture



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 Infra 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 OpenTofu modules

Vendor-Managed Runner Pools

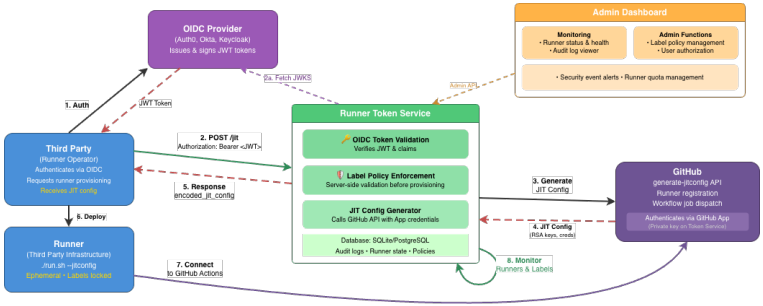
Challenges:

- Long lived admin credentials
- Runner metadata cannot be enforced
- Lack of central administration
- Lack of auditability

Proposed Solution:

- GHA Runner Token Service (GHARTS)
- Use LFID for authentication
- No Admin Credentials to vendors
- Ephemeral, non reusable credentials
- Enforcement of metadata and quota
- Centralized administration
- Auditability

GHARTS Provisioning Flow



- Vendor authenticates using existing credentials (LFID)
- Vendor requests a JIT config 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

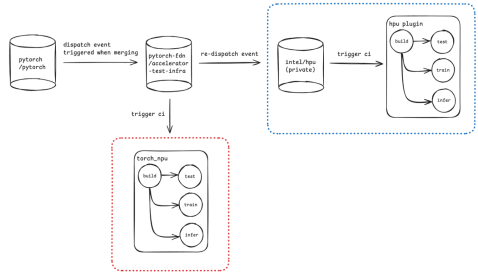
Vendor-Managed Test Systems

Accelerator Integration Working Group Initiative:

- Loosely coupled with PyTorch CI
- Events relayed to external repository
- Vendors provide test infrastructure and environments
- Out-of-tree test workflows

Use Cases:

- Specialized hardware testing (TPUs, custom accelerators)



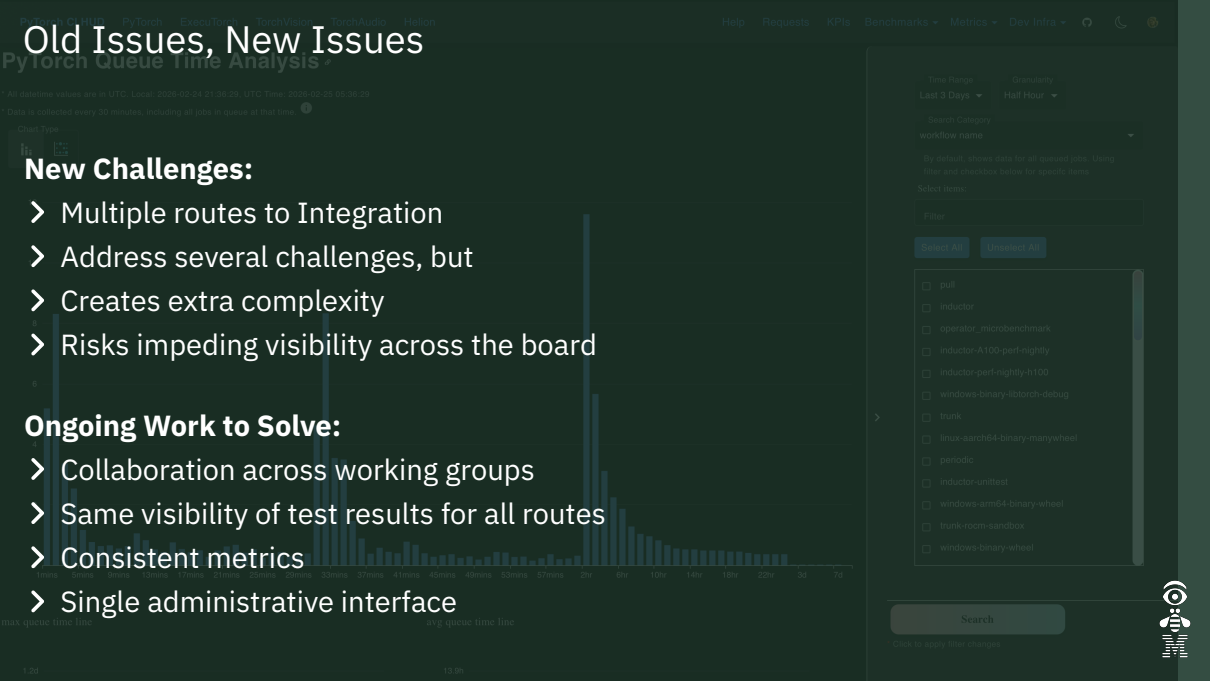
Old Issues, New Issues

New Challenges:

- Multiple routes to Integration
- Address several challenges, but
- Creates extra complexity
- Risks impeding visibility across the board

Ongoing Work to Solve:

- Collaboration across working groups
- Same visibility of test results for all routes
- Consistent metrics
- Single administrative interface



Results & Lessons Learnt

leonardo.ai, CC0

Fleet Composition

Onboarded:

- **Public Cloud Credits:** AWS, Meta
- **Vendor-Managed Runner Pools:** AMD ROCm, IBM s390x, IBM ppc64, Intel XPU, NVIDIA CUDA B200, NVIDIA Windows RTX
- **Vendor-Managed Test Systems:** Huawei Ascend NPUs, Intel Gaudi HPU
- ...and GitHub Hosted runners too!

In progress:

- **Vendor-Managed Runner Pools:** IBM s390x (via GHARTS)
- **Vendor-Managed Test Systems:** RedHat RHEL

Ongoing Work

Public Clouds

- Migrate CI Jobs (with the help of AI!)
- Migrate Autoscaler to ARC based system

Runner Pools

- Deploying GHARTS
- Onboarding IBM pools
- Integrate ARC with GHARTS

Test Systems

- Integrate into hud
- Enable PR comments and votes

Governance

- Finalize onboarding process
- Finalize SLO definitions

Lessons Learned

- **Clear onboarding:** Streamlined process and transparent decision making
- **Document SLOs:** Define roles and SLOs, include community engagement
- **Reusable Tech Stack:** Share proven configurations and best practices
- **Engage Vendors in WG:** Vendors can and want to contribute to the solution
- **Self Service First:** Distributed engineering effort
- **Empower Maintainers:** Give infra team tools to manage at scale
- **Security:** Consider contributed code and infrastructure as untrusted by default
- **No one-size-fit-all:** Multiple contribution paths reduce attrition

What’s Next

- Third-party builds and releases
- Consistent metrics, monitoring and UX across the fleet
- Adopt standards like OpenTelemetry/CDEvents
- Continue balancing community needs and project sustainability

To summarize

- PyTorch builds the LLMs that power the AI revolution
- Hosted by the PyTorch Foundation

CI/CD Infra challenges:

- community
- scale (financial/engineering)
- security

- The solution is a mixture of governance and technology
- Engaged with Vendor through Working Groups to address challenges

- Multiple paths to contribution, consolidate visibility and administration
- Work continues to make the CI/CD truly cloud agnostic

Questions?

Brecon Beacons, Wales, Photo by Carl Jorgensen, CC0

References & Contact

Resources:

- PyTorch: `https://pytorch.org`
- PyTorch : `https://github.com/pytorch/pytorch`
- PyTorch Foundation: `https://pytorch.org/foundation`
- Multi-Cloud Infrastructure WG: `https://github.com/pytorch-fdn/multicloud-ci-infra`
- Accelerator Integration WG: `https://github.com/pytorch-fdn/accelerator-integration-wg`

Contact:

- Andrea Frittoli
- andreafrittoli | afrittoli