THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

Embedded Linux Conference

Serverless for Open-Source Maintainers

Automating the Boring, Scaling the Impact

Hemant Bharadwaj & Antra Purohit · Microsoft

Open-Source Summit NA · Minneapolis · May 2026

Who should attend this talk

- Developers new to serverless concepts
- Open-source contributors and maintainers
- SRE / DevOps engineers exploring simple automation
- Students or beginners interested in cloud technologies
- Anyone curious about event-driven applications

Why This Talk & Take Away

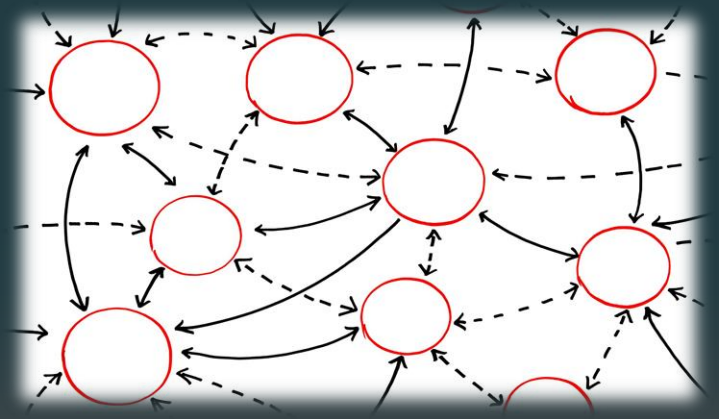
- **Maintainer Operational Challenges**
  - Maintainers face continuous operational tasks like triage, reviews, releases, and community moderation that consume significant time.
- **Serverless as Enabler**
  - Serverless computing reduces infrastructure management and scales cost-effectively, allowing maintainers to focus on design and mentorship.
- **Sustainable Automation Approach**
  - Automation is sustainable when portable via open event standards and observable through standardized telemetry.
- **Practical Outcomes for Attendees**
  - Attendees gain a roadmap for automation, a vendor-neutral architecture, and a playbook for incremental, safe automation adoption.

The Maintainer Problem: Why Toil Doesn't Scale

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA
Embedded Linux Conference

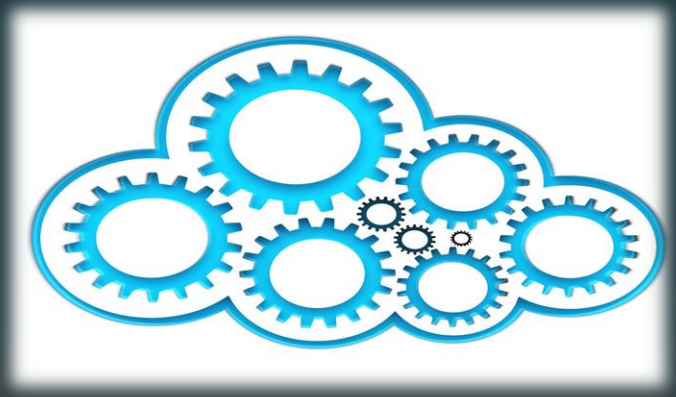
Scaling Challenges in Open Source

- **Community Interaction Growth**
  - As projects grow, maintainers face increasing community interactions including issues, pull requests, and discussions.
- **Operational Overhead Challenges**
  - Maintainers spend significant time on issue triage, pull request validation, and managing releases, causing burnout risks.
- **Need for Automation**
  - Scaling open-source projects without automation is unsustainable, requiring new serverless automation approaches to ease burdens.
- **Focus on Innovation**
  - Automation helps maintainers reclaim time to focus on innovation and community-building instead of repetitive tasks.



Where Maintainer Time Goes?

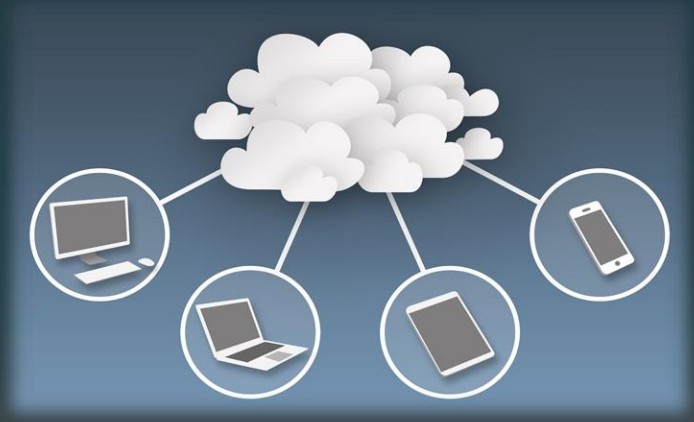
- **Issue Intake Management**
  - Maintainers spend significant time labelling, triaging, deduplicating, and guiding issues to support channels.
- **Pull Request Hygiene**
  - Ensuring pull requests meet standards, checks run, reviewers assigned, and contributors understand next steps.
- **Release Operations**
  - Tasks include changelog curation, version management, artifact signing, publishing, and communications post-release.
- **Community Operations**
  - Maintainers handle moderation, code of conduct issues, stakeholder notifications, and consistent communication.



Why Serverless?

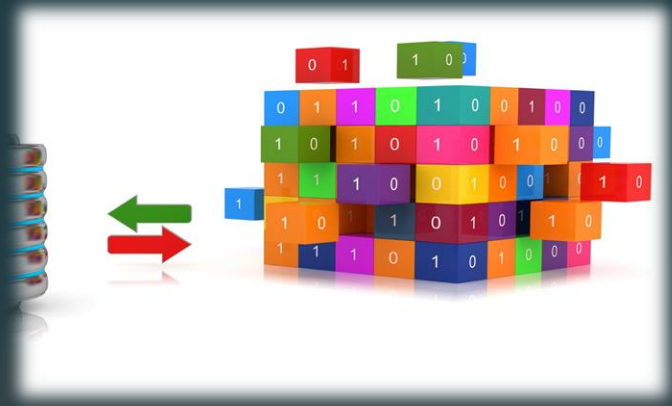
Benefits of Event-Driven Automation

- **Infrastructure-Free Automation**
  - Serverless computing removes infrastructure management, enabling scalable event-driven workflow automation without ongoing resource use.
- **Cost Efficiency and Elasticity**
  - Pay-as-you-go pricing and automatic scaling provide efficient resource use and consistent performance during workload spikes.
- **Seamless Integration and Portability**
  - Serverless functions integrate with APIs and cloud services, supporting portable, vendor-neutral automation pipelines for maintainers.
- **Responsive Open-Source Workflows**
  - Event-driven triggers automate tasks like issue labeling and release orchestration, reducing operational overhead in open-source projects.



Serverless From a Maintainer's Perspective

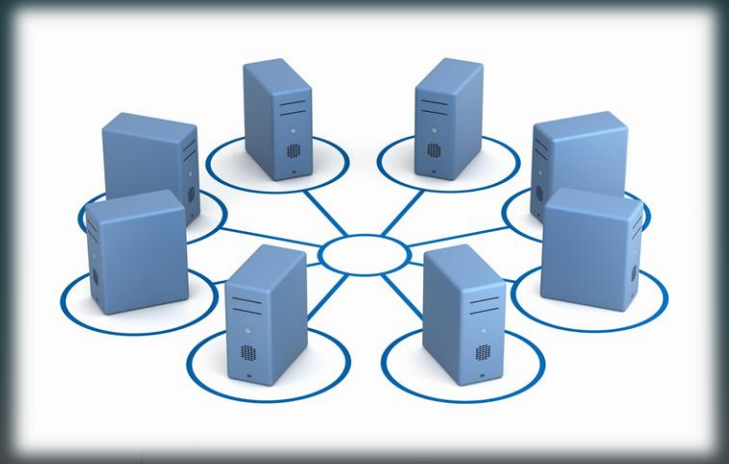
- **Event-Driven Compute**
  - Compute runs only when triggered by events like pull requests or CI completions, aligning with bursty maintainer workflows.
- **Scale-to-Zero Efficiency**
  - Serverless scales down to zero during idle times, reducing cost and resource usage for open-source projects.
- **Reduced Maintenance Burden**
  - Serverless reduces fleet management and uptime work by minimizing long-running services and simplifying deployments.
- **Modular and Replaceable Automation**
  - Automation is modular, allowing maintainers to disable functions individually and contributors to add handlers easily.



Use Case: Issue Triage

Automating Issue Management

- **Automated Issue Categorization**
  - Serverless functions use machine learning or rules to assign labels based on issue content, ensuring consistent categorization.
- **Stale Issue Detection**
  - Automation identifies inactive issues and applies actions like closing or flagging for review to maintain issue relevance.
- **Expertise-Based Issue Routing**
  - Issues are routed to maintainers based on expertise, improving response time and accountability.
- **Improved Contributor Experience**
  - Automation provides faster feedback and clearer communication, enhancing contributor engagement and project health.



Use Case: PR Validation

Enhancing Contribution Workflows

Automated Pull Request Validation

- Serverless functions automatically trigger tests and enforce code quality when pull requests are opened or updated.

Policy Checks and Feedback

- Automation enforces commit message format, documentation, and license compliance, providing immediate contributor feedback.

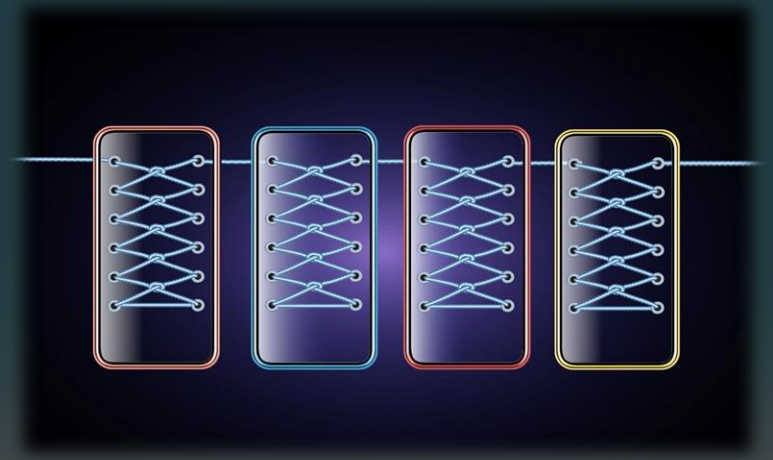
Human and Automation Collaboration

- Automated checks complement human review to ensure quality and efficiency in contribution workflows.

Use Case: Release Automation

Streamlining Release Processes

- **Automated Release Workflow**
  - Serverless automation orchestrates release steps, reducing manual effort and errors for reliable software delivery.
- **Changelog Generation**
  - Automatic changelog creation analyzes commit history and pull requests for accurate release documentation.
- **Community Notifications**
  - Automated notifications keep contributors and users informed of new releases to maintain engagement.
- **Observability Integration**
  - Monitoring release outcomes enables early failure detection, improving pipeline transparency and quality.



Observability

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA
Embedded Linux Conference

Monitoring and Insights

- **Importance of Observability**
  - Observability is essential for monitoring performance and diagnosing issues in automated systems.
- **Tools for Visualization**
  - Grafana and Application Insights provide dashboards to visualize latency, failure rates, and execution frequency.
- **Logging, Tracing, and Alerts**
  - Logging and tracing help debug issues; alerts notify of critical failures for timely intervention.
- **Designing for Observability**
  - Incorporating observability from design ensures system reliability and provides insights into project dynamics.

Common Pitfalls

Challenges in Automation

- **Balancing Automation and Control**
  - Over-automation can create complex systems that are hard to manage. Balance with manual intervention is essential.
- **Hidden Component Coupling**
  - Undetected dependencies between components reduce resilience and complicate updates in automated systems.
- **Vendor Lock-In Risks**
  - Relying on proprietary services may cause vendor lock-in; open standards help maintain portability.
- **Monitoring and Observability**
  - Lack of monitoring reduces automation effectiveness and hinders issue detection and resolution.

Key Takeaways

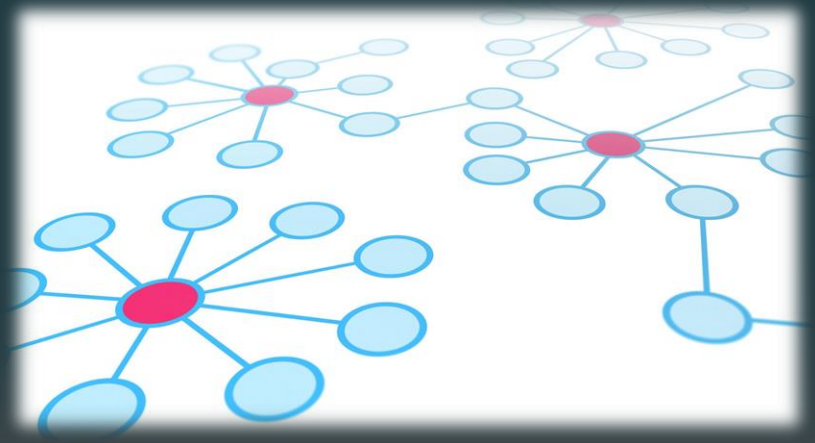
Summary of Insights

- **Automation Benefits**
  - Automating repetitive tasks reduces operational overhead and allows maintainers to focus on innovation and project growth.
- **Core Architectural Principles**
  - Event-driven design, portability, and observability create a strong foundation for scalable and efficient automation systems.
- **Avoiding Pitfalls**
  - Prevent over-automation and vendor lock-in to ensure sustainable, adaptable, and maintainable solutions for projects.
- **Community and Scalability**
  - Effective automation improves contributor experience, project reliability, and fosters a more engaged open-source community.

Conclusion

Closing and Next Steps

- Benefits of Serverless Automation
  - Serverless automation streamlines workflows, reduces manual efforts, and enhances project sustainability for maintainers.
- Starting Small and Expanding
  - Maintainers are encouraged to begin by automating one workflow and gradually increase automation based on measured impact.
- Community Collaboration
  - Sharing knowledge and collaborating within the community is key to adopting best practices and succeeding.
- Empowering Maintainers
  - Automation empowers maintainers to focus on building software and nurturing vibrant open-source communities.



THANK-YOU