Building Bazel Packages for AI/ML: SciPy, PyTorch, and Beyond

Conference Proposal

PyData Seattle 2025

Speakers:

Ramesh Oswal
Jiten Oswal

November 8, 2025

Contents

1 Proposal Overview

1.1 Title

Building Bazel Packages for AI/ML: SciPy, PyTorch, and Beyond

1.2 Speakers

- **Ramesh Oswal**
- **Jiten Oswal**

1.3 Session Duration

280 minutes (4 hours 40 minutes)

1.4 Target Audience

- Software engineers working with AI/ML infrastructure
- DevOps and MLOps practitioners
- Data scientists interested in reproducible builds
- Build system engineers
- Researchers managing complex dependency chains

2 Abstract

AI/ML workloads depend heavily on complex software stacks, including numerical computing libraries (SciPy, NumPy), deep learning frameworks (PyTorch, TensorFlow), and specialized toolchains (CUDA, cuDNN). However, integrating these dependencies into Bazel-based workflows remains challenging due to compatibility issues, dependency resolution, and performance optimization.

This session explores the process of creating and maintaining Bazel packages for key AI/ML libraries, ensuring reproducibility, performance, and ease of use for researchers and engineers.

3 Session Outline

3.1 Introduction to Bazel for AI/ML (20 minutes)

This section introduces participants to Bazel and its relevance for AI/ML workflows:

- Overview of Bazel build system fundamentals
- Why Bazel matters for AI/ML workloads
- Benefits of Bazel for dependency management

- Reproducibility and hermetic builds
- Cross-platform compatibility considerations
- Comparison with traditional Python packaging (pip, conda)

Learning Outcomes:

- Understand Bazel's core concepts and terminology
- Recognize the advantages of Bazel for AI/ML projects
- Identify use cases where Bazel adds value

3.2 Challenges in AI/ML Bazel Packaging (30 minutes)

An in-depth exploration of the unique challenges encountered when packaging AI/ML libraries with Bazel:

- **Transitive Dependencies:** Managing complex dependency graphs
- **Build System Differences:** Bridging CMake, setuptools, and Bazel
- **GPU Acceleration:** Integrating CUDA, cuDNN, and hardware-specific toolchains
- **Binary Compatibility:** ABI compatibility across different platforms
- **Large Binary Sizes:** Handling large ML model files and libraries
- **Version Conflicts:** Resolving conflicts between library versions
- **Native Extensions:** Compiling C/C++ extensions for Python libraries

Learning Outcomes:

- Identify common pitfalls in AI/ML library packaging
- Understand dependency resolution complexities
- Recognize GPU toolchain integration challenges

3.3 Strategies for Packaging (30 minutes)

Practical strategies and techniques for successfully packaging AI/ML libraries:

- **SciPy Stack:** Building NumPy, SciPy, and related numerical libraries
- **PyTorch:** Creating Bazel rules for PyTorch with CUDA support
- **TensorFlow:** Leveraging existing Bazel infrastructure
- **Dependency Management:** Strategies for managing transitive dependencies
- **Performance Optimization:** Ensuring optimal build and runtime performance
- **Cross-Platform Support:** Building for Linux, macOS, and Windows

- **Testing and Validation:** Ensuring package correctness and functionality

Learning Outcomes:

- Apply proven packaging strategies to common libraries
- Optimize builds for performance and compatibility
- Implement effective testing strategies

3.4 Best Practices for Distribution and Maintenance (20 minutes)

Guidelines for maintaining and distributing Bazel packages in production environments:

- **Version Management:** Semantic versioning and release strategies
- **Repository Structure:** Organizing Bazel workspaces effectively
- **Distribution Channels:** Using Bazel Central Registry and private registries
- **Documentation:** Creating user-friendly package documentation
- **CI/CD Integration:** Automating build and test pipelines
- **Community Engagement:** Contributing to open-source Bazel rules
- **Monitoring and Updates:** Keeping packages up-to-date with upstream changes

Learning Outcomes:

- Establish sustainable maintenance workflows
- Implement effective distribution strategies
- Integrate packages into CI/CD pipelines

3.5 Hands-on Demo (144 minutes)

An extensive practical demonstration where participants follow along:

- **Environment Setup:** Configuring Bazel for AI/ML development
- **Building SciPy:** Step-by-step package creation for SciPy
- **Integrating PyTorch:** Adding PyTorch with GPU support
- **Creating Custom Rules:** Writing Bazel rules for custom libraries
- **Dependency Resolution:** Handling complex dependency chains
- **Testing and Validation:** Running tests to verify package functionality
- **Performance Benchmarking:** Comparing build times and runtime performance
- **Troubleshooting:** Common issues and debugging techniques

Learning Outcomes:

- Build complete Bazel packages for real AI/ML libraries
- Debug and troubleshoot packaging issues
- Apply learned techniques to custom projects

3.6 Q&A and Open Discussion (36 minutes)

An interactive session for:

- Addressing participant questions
- Discussing specific use cases and challenges
- Sharing experiences and best practices
- Exploring future directions for Bazel in AI/ML
- Networking and community building

4 Key Takeaways

Participants will leave this session with:

1. **Practical Knowledge:** Hands-on experience building Bazel packages for AI/ML libraries
2. **Problem-Solving Skills:** Techniques to overcome common packaging challenges
3. **Best Practices:** Industry-standard approaches to package maintenance and distribution
4. **Reproducible Workflows:** Ability to create hermetic, reproducible build environments
5. **Community Resources:** Knowledge of community tools, resources, and support channels

5 Prerequisites

To get the most out of this session, participants should have:

- Basic understanding of build systems and dependency management
- Familiarity with Python and AI/ML libraries
- Basic command-line proficiency
- (Optional) Prior exposure to Bazel or similar build systems
- Laptop with Docker or Bazel installed (for hands-on portion)

6 Materials Provided

- Complete Bazel workspace examples
- Sample BUILD files for SciPy, PyTorch, and other libraries
- Documentation and reference guides
- Troubleshooting checklists
- Links to additional resources and community support

7 Speaker Information

7.1 Ramesh Oswal

Ramesh Oswal is an experienced software engineer specializing in build systems and AI/ML infrastructure. With extensive experience in Bazel and large-scale dependency management, Ramesh has contributed to numerous open-source projects focused on reproducible ML workflows.

7.2 Jiten Oswal

Jiten Oswal brings expertise in DevOps, MLOps, and infrastructure automation. His work focuses on creating efficient, scalable build pipelines for AI/ML applications, with a particular emphasis on cross-platform compatibility and performance optimization.

8 Why This Session Matters

As AI/ML workloads become increasingly complex and production-critical, the need for reliable, reproducible build systems has never been greater. Traditional Python packaging approaches often fall short when dealing with:

- Complex native dependencies (CUDA, cuDNN, MKL)
- Large-scale monorepo structures
- Strict reproducibility requirements
- Cross-platform deployment needs
- High-performance computing environments

This session addresses these challenges head-on, providing practical solutions that teams can implement immediately. By the end of the session, participants will have the skills and confidence to manage their own AI/ML dependencies using Bazel, leading to more reliable and maintainable systems.

9 Expected Outcomes

After attending this session, participants will be able to:

1. Create Bazel packages for popular AI/ML libraries
2. Integrate GPU acceleration toolchains into Bazel builds
3. Manage complex dependency graphs effectively
4. Set up reproducible build environments for AI/ML projects
5. Troubleshoot common packaging issues
6. Contribute to the Bazel AI/ML ecosystem

10 Contact Information

For questions or additional information about this proposal, please contact the speakers through the GitHub repository:

`https://github.com/RameshOswal/pydata-2025-conference-bazel-ai-pkgs`

Thank you for considering this proposal for PyData Seattle 2025!