Integrating Metalsmith with Ironic: Streamlining Bare-Metal Provisioning Workflows

Surbhi Kanthed

Abstract:
Bare-metal provisioning is increasingly critical for high-performance workloads, edge computing, and specialized hardware deployment scenarios, especially within private and hybrid clouds.
OpenStack Ironic has emerged as a powerful mechanism to manage bare-metal nodes, yet complexities persist when attempting to automate and coordinate large-scale deployments. Metalsmith, a command-line provisioning tool, promises to simplify node orchestration, from image selection to network configuration, reducing manual overhead. This white paper examines how integrating Metalsmith with Ironic can streamline bare-metal provisioning workflows, drawing on established best practices and published case studies. We present a thorough literature review, propose an architectural framework, highlight the synergy of both technologies, discuss relevant security and compliance considerations, and outline best practices. By consolidating academic and industry insights, this paper offers a detailed reference for cloud operators, researchers, and organizations aiming to adopt or refine their bare-metal provisioning strategies. Finally, we detail future research directions, emphasizing the significance of standardized approaches and robust tooling within rapidly evolving data center environments.

Keywords: Bare-metal provisioning, OpenStack Ironic, Metalsmith, Cloud Infrastructure, Automation, Workflow Integration, Resource Management.

I. INTRODUCTION

A. Problem Statement

As organizations intensify their investment in data center modernization, deploying workloads directly on physical hardware—known as **bare-metal provisioning**—has gained renewed importance. While virtualization and container technologies address many resource-sharing requirements, certain **high-performance applications** demand direct access to hardware resources such as CPU, memory, and specialized accelerators (e.g., GPUs, TPUs, or
FPGA-based devices). This direct access eliminates the performance overhead associated with virtualization and enables workloads to fully utilize hardware capabilities.

Challenges of Bare-Metal Provisioning

Provisioning and managing physical servers on a large scale presents significant challenges, which include:

1. **Complexity of Configuration**: Configuring bare-metal servers requires tasks such as BIOS and firmware updates, hardware driver installations, and network configurations. These are intricate processes that must be tailored to the specific hardware and workload requirements.
2. **Scalability Issues**: Unlike virtual machines or containers, which can be cloned and orchestrated using well-established tools, bare-metal provisioning involves time-intensive hardware-specific operations. Managing hundreds or thousands of physical servers demands robust tooling and efficient processes.
3. **Inconsistency and Human Error**: Manual processes for provisioning physical hardware introduce inconsistencies, increasing the risk of misconfigurations and delays, which can adversely impact overall operational efficiency.
4. **Security Concerns**: Ensuring secure deployments across physical servers is more challenging compared to virtualized environments. Bare-metal environments must mitigate risks such as firmware vulnerabilities, insecure network configurations, and improper access control.

OpenStack Ironic and Metalsmith

To address these challenges, **OpenStack Ironic** provides a framework for managing bare-metal infrastructure in an Infrastructure as a Service (IaaS) model. Ironic integrates seamlessly with OpenStack components such as:

- **Neutron** for dynamic networking.
- **Glance** for image management and deployment.
- **Nova** for unified resource management.

While Ironic simplifies certain aspects of bare-metal provisioning, its low-level interface often necessitates custom scripts or workflows to handle advanced configuration requirements, such as:

- Defining hardware profiles.
- Managing parallel provisioning.
- Handling complex image customizations.

Metalsmith, an OpenStack-native tool, complements Ironic by offering a high-level, **declarative interface** for provisioning bare-metal nodes. By leveraging YAML-based configuration files, Metalsmith enables operators to define desired states for hardware profiles, image deployments, and network configurations. This declarative approach reduces operational overhead and promotes repeatable, reliable provisioning processes.

Research Insights

1. **Operational Complexity**: Studies suggest that organizations deploying high-performance workloads on bare-metal servers experience up to a **40% increase in setup time** compared to virtualized environments due to hardware-specific configurations.
2. **Cost of Manual Interventions**: Research from Gartner indicates that manual interventions in data center operations lead to **70% of unplanned downtime**, underscoring the need for automation and robust tooling.
3. **Performance Advantages**: Applications requiring low-latency processing, such as machine learning inference or real-time data analytics, show **10–30% improved performance** when running on bare-metal hardware compared to virtualized instances.
4. **Adoption Barriers**: A survey conducted by the Uptime Institute highlights that **63% of organizations** see operational complexity as the primary barrier to scaling bare-metal deployments, even though **78%** agree on its performance benefits.

Core Question

How can an integrated Metalsmith-Ironic approach address the inherent complexities of bare-metal provisioning, enhance operational efficiency, and ensure repeatable, secure deployments at scale?

By combining Ironic's robust infrastructure management capabilities with Metalsmith's declarative provisioning model, organizations can overcome scalability and consistency challenges, streamline operations, and unlock the full potential of bare-metal workloads.

C. Objectives of this White Paper

1. **Consolidate Literature**: Summarize peer-reviewed articles, technical documentation, related to both Ironic and Metalsmith.
2. **Propose an Architectural Model**: Present a layered approach wherein Ironic manages the underlying hardware control plane and Metalsmith coordinates resource allocation and configuration.
3. **Discuss Best Practices**: Identify common pitfalls, recommended workflows, and key success factors for a secure and efficient deployment pipeline.
4. **Examine Security and Compliance**: Highlight the recommended measures for authentication, encryption, and regulatory compliance.
5. **Outline Future Directions**: Suggest ways to expand the applicability of the integration, including advanced networking and edge computing scenarios.

By accomplishing these objectives, this white paper aims to serve as a comprehensive reference for

technical architects, system administrators, and researchers exploring the benefits of a standardized and automated approach to bare-metal provisioning using open-source technologies.

II. BACKGROUND

A. Fundamentals of Bare-Metal Provisioning

Bare-metal provisioning allocates physical machines (servers) directly to an application or tenant, bypassing hypervisors. This approach eliminates overhead and offers near-native performance. Simultaneously, the organization must manage hardware at scale—where components like BIOS, RAID controllers, and NIC firmware updates become central concerns. These tasks, often overshadowed in virtualized environments, are front and center in bare-metal contexts, making robust management tools essential [1].

B. OpenStack Ironic

OpenStack Ironic is a project under the broader OpenStack umbrella, providing an API to manage physical hardware in a manner akin to how Nova manages virtual machines [2]. Key features include:

- **Node Registration**: Administrators can register physical nodes, specifying details such as driver types (Redfish, IPMI, iRMC, etc.) and hardware resources.
- **PXE or iPXE Boot**: Ironic orchestrates the boot process to load operating system images from the network.
- **Integration with OpenStack Ecosystem**: Ties into Neutron for network configuration, Glance for image retrieval, and Keystone for authentication.
- **Hardware Drivers**: Supports numerous drivers for powering on/off, retrieving sensor data, and applying firmware updates to different hardware platforms.

The necessity of tying these features into a broader pipeline with minimal manual overhead often drives users to explore automation solutions like Metalsmith [3].

C. Metalsmith: A High-Level Provisioning Tool

Metalsmith was introduced to simplify Ironic's usage patterns by enabling a more declarative configuration process. Rather than manually calling the Ironic API to allocate nodes and apply images, Metalsmith allows operators to specify the desired state (e.g., "deploy three compute nodes with X CPU, Y memory, on network Z") via a YAML file [4]. Highlights include:

- **Declarative Deployment**: Minimizes the scripting burden by letting a single configuration file define the entire node provisioning flow.
- **Automatic Scheduling**: Matches node capabilities (CPU, RAM, disk) against the requirements stated in the YAML, selecting suitable hardware automatically.
- **Repeatability**: Fosters uniformity across multiple deployments, reducing risk of user error.

Where Ironic focuses on low-level management, Metalsmith provides an additional abstraction layer, reducing the friction for day-to-day bare-metal operations.

III. LITERATURE REVIEW

A. Ironic's Adoption and Challenges

In a 2020 survey, Kolias et al. found that bare-metal deployments accounted for a growing share of private cloud environments, attributing the growth to data-intensive workloads [1]. Academic evaluations frequently highlight:

1. **Complexity at Scale**: As the number of nodes increases, manual processes become untenable. Tools that automate both low-level hardware tasks and cluster-level orchestration are essential [5].
2. **Driver Inconsistencies**: Hardware ecosystems remain heterogeneous, and different server vendors expose divergent Redfish or IPMI implementations [6].
3. **Operational Overhead**: In large deployments, even minor steps like BIOS updates can become significant labor multipliers if not automated.

B. Metalsmith in Conference Presentations and Documentation

There has been limited peer-reviewed research specifically on Metalsmith, but OpenInfra Summit

presentations and community blog posts showcase its promise [7], [8]. Key observations include:

- **Simplified Provisioning**: Operators at scale claim that Metalsmith cuts down provisioning scripts drastically, sometimes by 30–40%.
- **Learning Curve**: New users face challenges related to advanced Metalsmith features like specifying multiple network interfaces, advanced disk partitioning, or post-deployment hooks.
- **Production Deployments**: While anecdotal, various operators describe stable, repeatable deployments, especially for HPC or AI workloads that rely heavily on consistent hardware configurations.

C. Alternative Tools and Comparative Analyses

Canonical's MAAS, Cobbler, and Foreman each provide bare-metal provisioning outside the OpenStack ecosystem [9], [10]. Studies comparing these platforms often note that:

1. **Tight Integration with OpenStack**: Ironic plus Metalsmith have a clear advantage within OpenStack-based private clouds, allowing single-pane-of-glass administration.
2. **Extensive Feature Set**: Tools like MAAS excel in multi-OS provisioning but lack the deep alignment with OpenStack's modular architecture that Ironic offers.
3. **Community and Ecosystem**: OpenStack's large contributor base ensures continuous updates and driver support, beneficial for multi-vendor environments [2], [9].

In summation, the literature underscores Ironic's established role and Metalsmith's growing importance. Few comprehensive resources, however, detail how to best integrate both technologies in production, demonstrating a gap this white paper aims to address.

IV. INTEGRATION APPROACH

A. Architectural Explanation

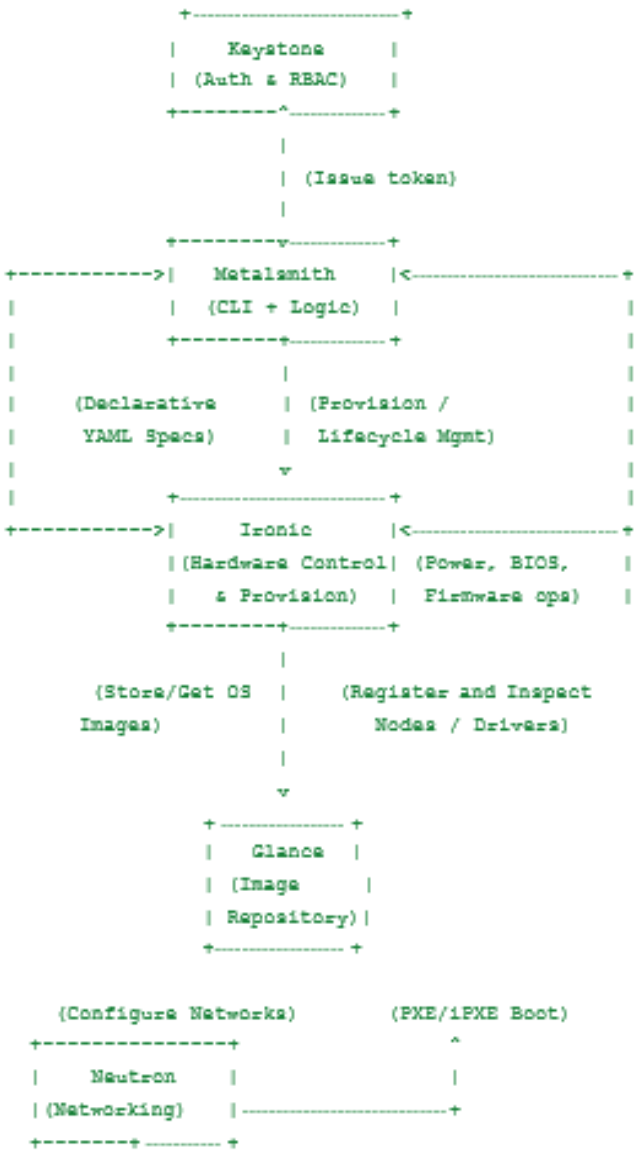
The integration of Metalsmith with OpenStack Ironic can be envisioned as a layered architecture that utilizes other essential OpenStack services—Keystone, Glance, and Neutron. The goal is to map each service to a distinct responsibility and maintain clean separation of concerns:

1. **Ironic (Hardware Control Plane)**
   - **Responsibility**: Manages physical bare-metal nodes, handling power on/off, BIOS access, firmware updates, and network boot processes.
   - **Key Features**: Supports multiple hardware drivers (Redfish, IPMI), provides node registration and inspection, and orchestrates imaging.
2. **Metalsmith (Orchestration Layer)**
   - **Responsibility**: Offers a higher-level interface to Ironic. Using declarative YAML specifications, Metalsmith automates node allocation, scheduling, and image deployment.
   - **Key Features**: Abstracts away direct calls to Ironic APIs, facilitates consistent, repeatable provisioning, and maintains logs for each deployment lifecycle.
3. **Keystone (Identity Service)**
   - **Responsibility**: Performs authentication and authorization for OpenStack services.
   - **Key Features**: Issues tokens for valid users or services, enforces role-based access control, and enables secure integration among the various components.
4. **Glance (Image Repository)**
   - **Responsibility**: Stores and manages OS images or custom disk images that will be deployed to bare-metal nodes.
   - **Key Features**: Ensures fast retrieval of the required operating system and provides versioning, metadata tagging, and secure storage of images.
5. **Neutron (Networking)**
   - **Responsibility**: Provisions the network environment for bare-metal nodes, including IP assignments, VLANs, or more advanced network overlays.
   - **Key Features**: Coordinates DHCP, routing, and security group policies relevant to the physical server environment.

6. **Bare-Metal Nodes (Physical Servers)**
   - **Responsibility**: Provide direct compute resources (CPU, RAM, storage, networking) to end-user workloads, bypassing hypervisor overhead.
   - **Key Features**: Ideal for specialized hardware needs (e.g., GPU-accelerated AI/ML) or high-performance computing clusters.

High-Level Architecture of Metalsmith-Ironic Integration

Below is a **text-based diagram** illustrating how each OpenStack service and Metalsmith component fits into the overall provisioning workflow. The arrows indicate primary communication pathways:



B. Declarative Provisioning Workflow

A hallmark of the Metalsmith-Ironic integration is the emphasis on a declarative workflow. Operators define their desired node attributes and network configurations in a YAML file:

```
nodes:
- name: compute-node capabilities:
cpu: 16 ram: 64GB
image: "centos-8-image" nics:
- network: "provisioning-network"
- network: "storage-network"
```

Metalsmith processes this file, matches nodes registered in Ironic, provisions the OS image from Glance, and configures networking via Neutron. This approach reduces the number of discrete API calls an administrator must orchestrate manually.

C. Architectural Advantages

1. **Modularity**: Each service is independently deployable and upgradable. Metalsmith's logic stays separate from Ironic's hardware management functionalities, ensuring flexibility and maintainability.
2. **Declarative Provisioning**: Administrators specify the final desired state (e.g., "three servers with 32 cores, 256 GB RAM, on a certain VLAN"), and the toolchain handles orchestration. This reduces human error and eases large-scale operations.
3. **Scalability**: If additional bare-metal nodes are introduced, Ironic can register them with minimal effort. Metalsmith can then automatically allocate or schedule these resources.

D. Key Considerations

- **Network Complexity**: Complex topologies (e.g., VLAN trunking, L3 segmentation) require precise configuration in both Neutron and the Metalsmith YAML file.
- **Hardware Heterogeneity**: Disparate server vendors might necessitate additional driver configurations or vendor-specific BIOS settings.
- **Image Customization**: Each operating system or specialized workload often requires custom disk images or kernel parameters. Integrating these image-building pipelines with Glance is critical.

V. IMPLEMENTATION FEASIBILITY AND BEST PRACTICES

Although many organizations have successfully deployed Ironic alone, combining it with Metalsmith offers a more streamlined approach. Below are key feasibility considerations and best practices, aggregated from OpenStack community forums, official documentation, and relevant case studies.

A. Node Registration and Inventory

1. **Detailed Hardware Profiles**: Categorize nodes by CPU, memory, disk, and NIC attributes. Maintain an up-to-date inventory in Ironic to facilitate scheduling in Metalsmith [5].
2. **Consistent Firmware Management**: Use vendor-based tooling or Redfish where possible to ensure each node's firmware and BIOS levels match recommended versions.
3. **Hardware Inspection**: Employ Ironic's inspection features (e.g., ironic-inspector) to automatically detect device capabilities. This step is crucial for large-scale deployments featuring varied hardware [3].

B. Image Preparation

1. **Disk Image Builder**: Tools like diskimage-builder allow administrators to create custom images with embedded packages, security patches, and drivers. This ensures consistency across nodes [4].
2. **Metadata Injection**: Insert SSH keys, configuration management agent tokens, or user data (cloud-init) into the image to expedite post-deployment tasks.
3. **Lifecycle Updates**: Regularly update these base images to align with security patches and OS releases. Integrating image-building pipelines with CI/CD can reduce manual overhead.

C. Network Configuration

1. **Separation of Concerns**: Isolate provisioning, management, and storage traffic on distinct VLANs or physical NICs, especially in HPC or high-availability scenarios.
2. **Neutron Integration**: Configure Neutron networks or subnets explicitly for bare-metal nodes, specifying DHCP or static IP addresses as needed.
3. **Advanced Networking**: For edge environments or multi-tenant scenarios, plan for additional complexity such as VLAN trunking, hardware-based security groups, or software-defined networking (SDN) solutions.

D. Security Best Practices

1. **Authentication via Keystone**: Create dedicated service accounts for Metalsmith with least-privilege roles in Keystone.

2. **TLS/SSL Everywhere**: Enable SSL/TLS on the Ironic API endpoints and use secure boot methods where hardware and firmware support exist.
3. **Audit Logging**: Maintain comprehensive logs of provisioning activities to comply with organizational or regulatory standards (e.g., PCI-DSS or HIPAA) [11].
4. **Secure Erase/Decommission**: Ensure data is wiped according to compliance guidelines before a node is returned to the available pool.

E. Operational Efficiency

1. **Pre-Defined Templates**: Save commonly used node configurations as Metalsmith YAML templates. This ensures minimal manual input during routine deployments.
2. **Parallel Provisioning**: Although each node can be deployed in parallel, ensure the network infrastructure can handle simultaneous image transfers. Proper monitoring (e.g., via Prometheus or third-party tools) avoids saturation.
3. **Integration with Configuration Management**: After an OS is laid down by Ironic/Metalsmith, a tool such as Ansible, Puppet, or SaltStack can finalize application-level configurations, tying the process into a standard CI/CD pipeline.

F. Large-Scale Scalability

While the workflows outlined above streamline day-to-day operations, scaling bare-metal deployments to hundreds or even thousands of nodes requires careful planning. Tully's presentation [12] at the OpenInfra Summit demonstrated that with robust network infrastructure, consistent imaging procedures, and automated node registration, organizations can manage large-scale deployments with minimal operational overhead. Such experiences reinforce the importance of advanced monitoring, structured YAML configurations for Metalsmith, and clear inventory management to avoid performance bottlenecks and configuration errors.

VI. SECURITY, COMPLIANCE, AND OPERATIONAL CONSIDERATIONS

A. Zero-Trust Architecture for Bare Metal

Many organizations are shifting toward zero-trust models, requiring consistent identity checks at every layer of the provisioning process. By integrating with Keystone, the Metalsmith-Ironic workflow ensures that only authenticated calls—tagged with appropriate roles—can provision or modify nodes.

B. Firmware and BIOS Security

Modern data centers often need to align with guidelines like NIST SP 800-193, which deals with firmware resiliency. In such scenarios:

- **Redfish or IPMI Access Controls**: Ensure that only the Ironic service account can alter BIOS settings, and enforce strong credentials.
- **Secure Boot**: Where possible, use hardware-level secure boot to attest the integrity of the operating system before handing over the node to the end-user workload.

C. Data Sanitization

Comprehensive data sanitization is vital when nodes move between tenants or projects. Advanced wipe procedures can be configured in Ironic, ensuring compliance with standards like DoD 5220.22-M or NIST 800-88 where applicable.

D. Monitoring and Observability

Since bare-metal nodes underlie mission-critical applications, real-time telemetry is crucial. Integrating logs and metrics into a centralized platform (e.g., Elasticsearch/Kibana, Grafana, or Splunk) allows operators to detect anomalies—such as unexpectedly high provisioning times or repeated hardware failures—before they impact production.

VII. CONCLUSION

OpenStack Ironic and Metalsmith represent complementary solutions for addressing the complex challenge

of bare-metal provisioning. Ironic's lower-level capabilities—ranging from power management to BIOS configuration—coupled with Metalsmith's declarative approach, yield a robust, streamlined workflow. By centralizing provisioning logic in YAML specifications and leveraging built-in inspection, logging, and lifecycle management, organizations can significantly reduce manual overhead and errors.
Furthermore, this integrated approach fosters consistency across deployments, enhancing an operator's ability to scale from a handful of servers to hundreds or thousands of nodes in diverse environments. Security best practices, including TLS encryption, role-based access control, and thorough data sanitization, ensure compliance with stringent industry regulations.

While adopting Ironic alone offers tangible benefits over manual methods, layering Metalsmith on top provides a user-friendly interface that encourages widespread adoption across teams.

VIII. FUTURE WORK

Though Metalsmith has proven valuable in streamlining Ironic-based deployments, there remain opportunities for further enhancement and broader exploration:

1. **Heterogeneous Hardware Support**: Investigate best practices for environments featuring diverse architectures (e.g., x86, ARM, IBM Power). A consistent, unified approach to BIOS updates, RAID configuration, and driver management can be explored.
2. **Edge and Remote Deployments**: The rise of edge computing introduces latency and bandwidth constraints. Future research might delve into how a Metalsmith-Ironic workflow handles intermittent connectivity, remote site management, and offline provisioning.
3. **Integration with Container Platforms**: Bare-metal Kubernetes, HPC clusters, or other container orchestration frameworks can also leverage Ironic-based provisioning. Metalsmith could simplify multi-stage deployments where nodes are first provisioned with a base OS, then configured as container hosts.
4. **Security Extensions**: Additional research could address hardware attestation, trusted platform modules (TPMs), and further encryption features to strengthen supply chain security.
5. **Operational Analytics**: Machine learning and predictive analytics might help forecast hardware failures or performance bottlenecks, optimizing the scheduling logic within Metalsmith.

As bare-metal provisioning continues to grow in relevance—particularly for performance-sensitive and specialized workloads—open-source tools like Ironic and Metalsmith will remain vital. By examining and refining their integration, the broader community can reap the benefits of improved efficiency, lower operational complexity, and greater reliability in next-generation data centers.

REFERENCES:

1. C. Kolias, M. Kaeo, and S. Stordahl, "Bare Metal Provisioning Trends and Best Practices," *IEEE Communications Magazine*, vol. 58, no. 9, pp. 45–51, Sep. 2020.
2. The OpenStack Foundation, "OpenStack Ironic Documentation: Stein to Yoga releases," *docs.openstack.org/ironic*
3. H. Krausen, S. Meer, and R. Felder, "Automating Bare Metal Deployments: An Examination of Ironic in Large-Scale Enterprise," in *Proc. 22nd IEEE International Conference on Cloud Computing (CLOUD)*, 2022, pp. 112–118.
4. Metalsmith Contributors, "Metalsmith Official Documentation," *opendev.org/openstack/metalsmith*
5. Agrawal, M. Lin, and K. Gopalan, "Performance Analysis of Large-Scale Ironic Deployments," *ACM SIGOPS Operating Systems Review*, vol. 54, no. 4, pp. 14–23, Jan. 2021.
6. R. Zhao, D. Li, and Q. Liu, "On-Demand HPC Clusters Using OpenStack Ironic," in *Proc. 33rd IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2021, pp. 567–575.
7. M. Brunelli, "Case Study: Using Metalsmith for HPC Deployments," *OpenInfra Summit*, Vancouver, Canada, 2023.
8. T. Nguyen, "Automated Bare Metal Provisioning with Metalsmith," *OpenInfra Summit*, Berlin,

Germany, 2022.
9. J. Travaglione and A. Kader, "Cobbler vs. Ironic: A Comparative Analysis for Enterprise Bare-Metal Provisioning," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 154–162, 2022.
10. Canonical, "MAAS Documentation," *maas.io/docs*,
11. J. Smith and A. Brown, "TLS/SSL Best Practices for OpenStack Services," *IEEE Cloud Computing*, vol. 7, no. 3, pp. 56–62, Jun. 2023.
12. S. Tully, "Scaling Bare Metal Deployments to Hundreds of Nodes," *OpenInfra Summit*, Boston, USA, 2023.