Integration of Slurm and Kubernetes for University Research Computing Workloads

A Thesis Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Victor Hazlewood
December 2025

Dedication

I dedicate this work to my wife, Dr. Hollie Raynor, and to my son, Samuel Darius Hazlewood, who have been a wonderful source of love and support.

Acknowledgments

I would like to thank Dr. Greg Peterson who mentored and guided my work in preparing, developing, and completing this thesis. Thanks also to Matthew Bachstein for his invaluable support including installation and support of the pilot computing environment. Special thanks to my wife, Dr. Hollie Raynor, for her support, partnership, and encouragement across the period of time it took to complete all the coursework and this thesis.

Abstract

Slurm is the de facto standard for resource and workload management in various academic and research environments, particularly in university high-performance computing (HPC) clusters. Kubernetes is a cloud-native, open-source software system for managing resources and workloads of containerized applications. It is considered the de facto standard for artificial intelligence applications and orchestrated workflows. Conventional and accelerated computational resources, as well as storage resources, are widely available in university HPC clusters, along with Slurm for managing and scheduling jobs on these clusters. K8s services are typically absent from university HPC clusters. With increasing interest among university researchers in developing and using artificial intelligence applications using university HPC cluster resources and the prevalence of artificial intelligence and machine learning applications being developed in Docker containers and deployed on the widely adopted Kubernetes platform, there is a need to investigate solutions for the integration of Slurm and Kubernetes onto university HPC clusters to facilitate artificial intelligence workloads. An optimal solution would eliminate the need to bifurcate university HPC cluster resources into separately managed Slurm and Kubernetes clusters or eliminate the requirement to purchase and create an entirely new cluster solely for Kubernetes operation. This work describes the general artificial intelligence research use cases and computational needs of the university researcher, the differences between compute clusters managed by Slurm and Kubernetes, surveys the available Slurm and Kubernetes integration solutions, and describes the experiences of implementation and use of one of the integration solutions at the University of Tennessee, Knoxville, on a subset of resources on one of the university's HPC clusters.

Table of Contents

List of Tables

List of Figures

List of Code Listings

Chapter 1
Introduction

1.1 AI as a New Industrial Revolution

Interest and advancements in artificial intelligence (AI), particularly surrounding pre-trained language models, have accelerated over the last decade (Chang et al. (2024); Zhao et al. (2023); Wei et al. (2022); Bommasani et al. (2021)). The Executive Office of the President of the United States (POTUS) has issued several policy documents that emphasize the significance of AI for the nation. Under the Obama administration, the "Preparing for the Future of Artificial Intelligence" (POTUS (2016b)) and "The National Artificial Intelligence Research and Development Strategic Plan" (POTUS (2016a)) were issued in October 2016. The 2017 Trump administration in February 2019 issued Executive Order 13859 "Maintaining American Leadership in Artificial Intelligence" (POTUS (2019a)) establishing the American Artificial Intelligence Initiative, issued "The National Artificial Intelligence Research and Development Strategic Plan: 2019 Update" (POTUS (2019b)), and the 2025 Trump Administration in July 2025 issued "America's AI Action Plan" (POTUS (2025)). These documents issued by the POTUS over the last decade underscore the importance of AI for the nation and in "America's AI Action Plan" it spells out three pillars for the action plan of "innovation, infrastructure, and international diplomacy and security" and states that there is potential for AI to develop "An industrial revolution, an information revolution, and a renaissance—all at once" (POTUS (2025)). Additionally, Jensen Huang, CEO of NVIDIA Corporation, summarized key technology innovations in his keynote speech

at the 2024 NVIDIA GTC Conference including that in 2016 he handed the first accelerator-based 170 teraflops DGX-1 supercomputer to a start-up company called OpenAI, in 2017 the transformer arrived - a reference to the AI transformer architecture developed at Google (Vaswani et al. (2017)), 2022 marked the arrival of ChatGPT, and in 2023 the arrival of generative AI (Huang (2024a)). In wrapping up his keynote, Jensen Huang stated that we are in a "new industrial revolution" being built using "AI factories in data centers" that provide "a new way of doing [developing] software with generative AI" to develop new applications using "AI foundries" (Huang (2024b)). These important national strategic plans, AI-related technological developments, and the potential for transformational change across many disciplines and industries (Marr (2019); Ahmed et al. (2022); Chang et al. (2024)) have led to increasing interest from academia and industry in researching, developing, and employing many new forms and variations of AI technologies across many disciplines.

1.2 AI/ML DevOps with Kubernetes

Kubernetes is more than ten years old and is in use by nearly all public cloud providers, including Amazon Web Services (AWS), Microsoft Azure (Azure), Google Cloud Platform (GCP), and Oracle Cloud Infrastructure. Kubernetes, originally designed and developed by Google in June 2014 (Poulton and Joglekar (2025a)), is a cloud-native, open-source software system for managing resources and workloads of containerized application workloads. It is considered the de facto standard for AI applications and orchestrated workflows in cloud environments (Poulton and Joglekar (2025a); Wu et al. (2024)). Kubernetes abstracts the infrastructure of resources and services in a public or private cloud, making them easier to use. Kubernetes orchestrates the execution of Docker containers, virtual machines (VMs), web assembly applications (Wasm apps) (Poulton and Joglekar (2025b)), other services, and various applications wrapped in a Kubernetes Pod (more on Pods in Section 3.2), which are all generally referred to as Kubernetes microservices. This capability has powered scalable web-based corporate e-commerce enterprises in the cloud, including Amazon's retail website on AWS (Miller (2016); Black (2013)), Netflix on AWS (Amazon Web Services (2016)), eBay using a hybrid cloud approach including Azure and GCP (Arjun (2025)), and Adobe's Adobe

Experience Cloud, Adobe Creative Cloud, and Adobe Document Cloud, all hosted in Azure (Microsoft Corporation (2016)). Along with e-commerce and enterprise applications, many AI and machine learning (ML) applications are developed in Docker containers and can be orchestrated by Kubernetes. Many AI and ML (AI/ML) projects employ DevOps (Ebert et al. (2016)) in DevOps teams - teams that combine the work of application developers and operations staff - to streamline the software development lifecycle with many using Docker containers for their AI/ML software projects and providing for deployment with Kubernetes into cloud environments (Ebert et al. (2016)). For example, OpenAI utilizes Docker containers for application development and employs Kubernetes extensively for AI/ML application DevOps, as seen in applications such as ChatGPT, GPT-4, and DALL-E (Raj (2024)). NVIDIA Inference Microservices (NIM) provides a set of prebuilt microservices in containers designed to accelerate the deployment of AI large language models (LLM) and AI/ML applications with Kubernetes on NVIDIA DGX systems and cloud infrastructure (NVIDIA (2024a,b)). Additionally, the Hugging Face Hub (Hugging Face (2020)) is a web-based platform that facilitates collaborative open source AI/ML research, development, and experimentation and the hub provides models, datasets, and applications (applications are called "Spaces" on the Hub) shared by the community, many of which are developed in Docker containers and can be easily deployed using Kubernetes (Raina et al. (2025)).

With the release of the seminal research paper "Attention is All You Need" from Vaswani et al. (2017) in December 2017 and its revolutionary paradigm shift impact on AI/ML with the transformer deep learning architecture which "*underpins most applications of AI in development*" (Murgia (2023)), there is strong interest and popularity of AI/ML across academia and industry (Chang et al. (2024)). The development of new AI/ML technologies is driving technological innovations across various industries today (Ahmed et al. (2022)). This impact includes the development of ChatGPT based on GPT-3.5 and released in November 2022 (Marr (2023)), where the "T" stands for "Transformer" in Generative Pre-trained Transformer (GPT) (Radford et al. (2018)). OpenAI released GPT-5 in August 2025 (OpenAI (2025)).

Following this global trend in AI popularity, the development and use of AI/ML applications that can be implemented in containers using Kubernetes on campus cluster

computational resources can be expected to increase. In discussions with University of Tennessee, Knoxville (UTK) researchers and with university research leaders at public universities in Tennessee at the 2024 AI Across Tennessee Symposium (UTK OIT HPSC (2024)), the requests for capabilities that can run applications in Docker containers, specifically Kubernetes, have increased over the last two years. The running of containers converted to Apptainer files (.sif) at UTK on the central high-performance computing (HPC) cluster has been steady from August 2023 to August 2025 with a slight monthly average increase over the last twenty-four months, as shown in Figure 1.1. In discussions with leaders at a Southeastern Conference (SEC) research director's meeting in July 2025, the University of Florida and the University of Missouri indicated that they have begun investigating Kubernetes environments for computational research purposes (Erik Deumens (host) (2025)). Email discussions with Texas A&M University staff have indicated that they are also investigating Kubernetes, utilizing two three-node Kubernetes clusters. It is notable that the University of Florida's HiPerGator 4 is the most powerful university research cluster known, built with NVIDIA DGX hardware including B200 GPUs (University of Florida (2025a)). The interest in AI/ML research and training, particularly with containers and Kubernetes, is anticipated to continue increasing posing the question: "Is university research cyberinfrastructure prepared for this change in researcher modality?"

1.3 Slurm in University Research and Academics

Many universities have on-premises, general-purpose HPC cluster capabilities that use the open-source and freely available Slurm workload manager to support computational research for their faculty, staff, and student researchers. This includes: UTK (UTK OIT (2022)); many of the public universities in the State of Tennessee including University of Tennessee, Chattanooga (University of Tennessee, Chattanooga (2025)) and University of Memphis (University of Memphis (2023)); and all of the universities that are members of the SEC including Auburn University (Auburn University (2025)), University of Alabama (University of Alabama (2025)), University of Arkansas (University of Arkansas (2025)), University of Florida (University of Florida (2025b)) (including Slurm on HiPerGator 3

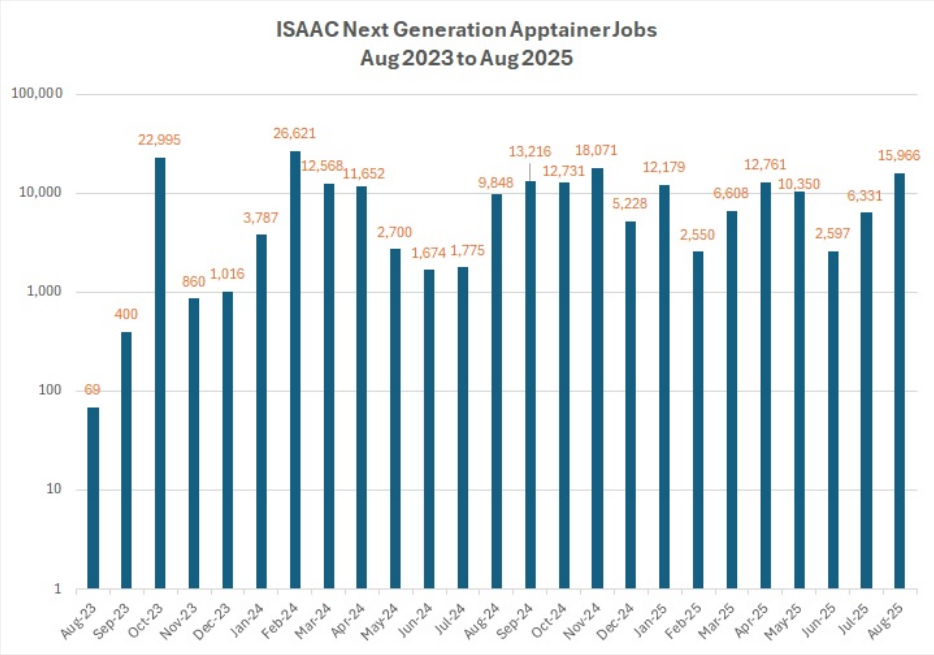

Figure 1.1: UTK HPC Cluster Apptainer Jobs

which is the most powerful supercomputer available at a U.S. university, is available for the researchers at Florida, and is not funded by the National Science Foundation), University of Georgia (University of Georgia (2025)), University of Kentucky (University of Kentucky (2025)), Louisiana State University (Louisiana State University (2025)), Ole Miss (University of Mississippi (2025)), Mississippi State University (Mississippi State University (2025)), University of Missouri (University of Missouri (2025)), University of Oklahoma (University of Oklahoma (2025)), University of South Carolina (University of South Carolina (2025)), UTK (UTK OIT (2022)), University of Texas, Austin (University of Texas (2025)), Texas A&M University (Texas A&M University (2025)), and Vanderbilt University (Vanderbilt University (2025)). A summary of HPC cluster resources from the SEC universities available to their respective researchers and each university's NSF HERD research expenditure for fiscal year 2023 (NSF (2024)) is shown in Table 1.1.

Except for the University of Florida, the University of Missouri, and Texas A&M University investigating Kubernetes as mentioned previously, and UTK investigating Kubernetes with this work, Kubernetes is not known to run in production on any of the general-purpose clusters listed in Table 1.1. Slurm is the de facto standard for resource

Table 1.1: SEC HPC Cluster Resources

University	Cluster	Vendor/ Integrator	Compute Nodes	Core Count	GPU Nodes	GPUs	GPU Vendor	Storage TBs/PBs	Storage Type	Resource Mgmt	HERD FY23 $M
Auburn University	Easley	Dell	392	15,100	13	30	NVIDIA	4 PB	GPFS	Slurm	$349
University of Alabama	UAHPC	Dell	92	3,840	6	10	NVIDIA	1.5 PB/(5.5 PB)	Panasas/(Other)	Slurm	$185
University of Alabama	CHPC	Dell	31	1,920	5	5	NVIDIA		Panasas/(Other)	Slurm	
University of Arkansas	Pinnacle	Dell/HPE	338	17,000	67	137	NVIDIA	5.5 PB/4.5 PB archive	Lustre/NFS/ZFS	Slurm	$222
University of Florida	HiPerGator	Lenovo/Mark III	512	60,000	263	1,104	NVIDIA	40 PB	Lustre	Slurm	$1,330
University of Georgia	Sapelo2	Dell	530	51,060	63	215	NVIDIA	12.5PB, 16PB	Lustre, ZFS	Slurm	$571
University of Georgia	Teaching	Dell	12	704	4	10	NVIDIA	300TB	ZFS	Slurm	
University of Kentucky	LCC	Dell/HPE	204	8.656	36	144	NVIDIA	3PB	GPFS	Slurm	$504
Louisiana State University	SuperMike-3	Dell	183	11,712	8	32	NVIDIA	Shared 2 PB	Lustre	Slurm	$384
Louisiana State University	SuperMIC	Dell	360	7,200	360	28	Intel Phi		Lustre	Torque/Moab	
Louisiana State University	Deep Bayou	Dell	13	624	13	30	NVIDIA		Lustre	Slurm	
University of Mississippi	Maple	Cray	140	5,454	37	45	NVIDIA	-	Lustre	PBS	$172
University of Mississippi	Magnolia	HPE	50	2,304	6	12	NVIDIA	-	Lustre	Slurm	
Mississippi State University	Orion	Dell	1,800	72,000	0	0	-	30 PB (shared)	Lustre	Slurm	$320
Mississippi State University	Hercules	Dell	512	40,960	0	0	-			Slurm	
Mississippi State University	Atlas	Cray	245	12,160	8	64	NVIDIA	2 PB, 6 PB	Lustre, VAST	Slurm	
University of Missouri	Hellbender	Dell	219	24,294	28	100	NVIDIA	8 PB	GPFS + VAST + Ceph	Slurm	$462
University of Oklahoma	Schooner	Dell	900	34,000	56	135	NVIDIA	2 PB	CephFS, NFS on ZFS	Slurm	$437
University of South Carolina	Hyperion III	Dell/HPE	356	16,616	1	8	NVIDIA	1.4 PB		Slurm	$262
University of South Carolina	Theia		38	3,808	10	40	NVIDIA	1.5 PB		Slurm	
University of Texas	Lonestar6	Dell	560	71,680	88	260	NVIDIA	X PB/8 PB	Lustre/BeeGFS	Slurm	$1,036
University of Tennessee	ISAAC NG	Dell	216	16,632	33	91	NVIDIA	4.5 PB & 800TB+6PB	Lustre & Lustre+DMF	Slurm	$339
University of Tennessee	Secure Enclave	Dell	60	2,848	2	4	NVIDIA	1.3 PBs	Lustre	Slurm	
Texas A&M University	Launch	Dell	45	8,640	10	20	NVIDIA	2 PB	ZFS	Slurm	$1,278
Texas A&M University	Grace	Dell	940	45,376	132	280	NVIDIA	5 PB	Lustre	Slurm	
Texas A&M University	New ($45M)	WWT	TBD	TBD	TBD	760	NVIDIA	TBD	TBD	TBD	
Vanderbilt University	ACCRE	Dell	750	20,100	80	320	NVIDIA	4.7PB/2PB/ 1.8PB/27PB	PanFS/Auristor/ GPFS/LStore	Slurm	$1,253

and workload management for a variety of academic and research applications on HPC clusters in many university environments (Wu et al. (2024) and shown in Table 1.1). All of these HPC clusters that employ the Slurm workload manager utilize Slurm to manage and allocate computational, storage, and other resources for queuing, managing, and running computational work to benefit their researchers. Currently, Slurm lacks native capabilities to run computational work composed of Docker containers. However, tools like Apptainer (Linux Foundation (2021)) can be easily integrated into Slurm to run computational work using Docker containers. Apptainer was formerly known as Singularity until November 2021. Many researchers are using HPC clusters for AI training workloads (Wickberg (2025)), and the use of Apptainer on the UTK HPC cluster is growing. A more detailed architecture and use overview of Slurm is described in Section 3.1.

In addition to university research, Kubernetes may play a role in academic courses. A review of the Fall 2025 academic courses at UTK that have Artificial Intelligence, AI, Machine Learning, or ML in the course title found that twenty-one courses match that search criteria, with 828 students registered on August 27, 2025. Those courses were in the Artificial Intelligence, Cinema Studies, Computer Science, Educational Leadership/Policy

Studies, English, Material Science, Mathematics, Philosophy, and Sociology departments. The complete list of UTK Fall 2025 courses and information collected is listed in Appendix B. Courses in the past with AI or ML subject matter requested reserved compute resources from the university HPC cluster, including the Physics 494 course, Introduction to Machine Learning, in Spring 2023, 2024, and 2025 (HPSC (2025)).

1.4 Kubernetes in University Research

As mentioned previously, Kubernetes was originally designed and developed by Google and is a cloud-native, open-source software system for managing resources and workloads of containerized application workloads. Kubernetes is considered the de facto standard for AI applications and orchestrated workflows. As discussed in section 1.2, AI/ML tools and applications are being developed using Docker containers, and many are being developed to be deployed using Kubernetes. A review of the HPC cluster documentation from SEC universities reveals that most of these campus HPC clusters utilize Slurm for workload management, with many centers implementing computational work developed in Docker containers using Apptainer. A review of all of the documentation available from the respective university computing centers referenced in section 1.3 indicates Kubernetes is not available on the on-premises centrally managed campus HPC clusters available from university research computing environments at UTK, at the public universities across Tennessee, nor from the Universities that are members of the SEC except for investigation of Kubernetes by University of Florida, University of Missouri, and Texas A&M University mentioned previously. This does not discount the strong likelihood that Kubernetes exists in faculty research laboratories not affiliated with a central campus HPC environment. Known examples of Kubernetes installations on university campus clusters include Purdue University's composable subsystem environments on the Anvil (Purdue University (2020b)) and Geddes clusters (Purdue University (2020d)), as well as Indiana University's Jetstream2 cluster (Indiana University (2020b)). All of these systems, Anvil, Geddes, and Jetstream2, are located on their respective campuses but are also resources available to the national community and are funded by the Office of Advanced Cyberinfrastructure at the National

Science Foundation (NSF) (Purdue University (2020a,c); Indiana University (2020a)). There are other cloud environments available for research, including the NSF-funded CloudLab (Duplyakin et al. (2019); CloudLab (2025)) and Chameleon (Keahey et al. (2020); Chameleon (2025)), which are both supported by the NSF Division of Computer and Network Services and were purpose-built to support computer science systems research, not general campus research computing. Chameleon only supports OpenStack. CloudLab supports any standard cloud software stack, including OpenStack, Hadoop, and Kubernetes. OpenStack and Kubernetes are distinct but complementary.

1.5 Use Cases

There is an ever-increasing variety of applications for AI including: literature summarization, data analysis and interpretation, writing assistance, code generation, text and voice translation, image recognition, image and video generation, synthetic data creation, and many others. Two key reasons explain why AI is thriving and showing accelerating interest and growth in research across many disciplines: tremendous amounts and types of digital data are available and access to a vast amount of data storage and computation is available (Marr (2019), p. 5). Digital data is available for research from a vast array of sources. The storage and computational resources are available from public cloud providers and from university cyberinfrastructure as shown in Table 1.1. These two key components, data and computation, combined with readily available AI software and tools from repositories, such as, Hugging Face and NVIDIA's NIM, provide extraordinary capabilities for university researchers. Generative AI and Natural Language Processing (NLP) models are two types of models that require the data and computational resources to turn these models into specific or general-purpose applications. Two significant computational use cases make up the process to develop a model and make it available for use: model training and model inference.

AI model training and AI model inference use cases have the potential to allocate and/or consume significant amounts of computational resources that university computational research providers should plan for: model training and model inference. Training an AI model

is generally the first step for developing an AI model. AI model training is a computationally intensive and iterative process that uses one or more algorithms and training data to modify the algorithm's coefficient values, accept feedback to correct errors, and create, as the output of the AI training, an AI model that will make accurate predictions based on the training. In general, AI model training, with the exception of the largest large language models (LLMs), is a one-time bounded computational expense that is complete once the model is trained and meets its acceptable model error rate. Once the AI model is trained, it can then be used for its designed purpose using AI model inference. AI model inference is the implementation of the model by processing new data and generating a prediction/output based on the training. AI inference requires computational resources that are low-latency and high-throughput, especially for real-time results. AI inference will usually require allocation of computational resources to be available for model throughput.

1.6 Contributions

Conventional and GPU compute, storage, and networking resources are widely available in university HPC clusters, along with Slurm to manage and schedule the jobs on those HPC clusters. With increasing researcher interest in developing AI applications using university HPC cluster resources and the prevalence of AI/ML applications being developed in Docker containers with Kubernetes deployment expectations, there is a need to investigate solutions for integrating Slurm and Kubernetes onto campus HPC clusters to facilitate AI/ML workloads. Of course, one solution would be to build a separate Kubernetes cluster for this research modality. However, an optimal solution would eliminate the need to build a new cluster for running Kubernetes or for bifurcating university HPC cluster resources into separately managed Slurm and Kubernetes clusters just for adding Kubernetes operation. This thesis describes the needs of AI/ML researchers, compares the differences between compute clusters managed by Slurm and Kubernetes, surveys available integration solutions for Slurm and Kubernetes, and discusses the selection and experience with implementing and using one of these integration solutions, specifically Slinky on the UTK HPC cluster.

The remainder of this thesis is organized as follows. Chapter 2 describes Slurm and Kubernetes integration possibilities, provides a literature review of available software tools that can integrate Slurm and Kubernetes, and provides information on Apptainer for running Docker containers on Slurm which can offer a partial solution to the problem. Chapter 3 provides an overview of Slurm and Kubernetes, describes how Slinky can satisfy the integration between Slurm and Kubernetes, and provides an overview of Slinky and its selection as a pilot implementation. Chapter 4 describes UTK's ISAAC Next Generation HPC cluster environment, the pilot deployed using UTK's HPC cluster, and the software installed and tests performed to investigate SchedMD's Slinky, which provides Slurm and Kubernetes on the same cluster. Chapter 5 presents conclusions and discusses future work.

Chapter 2
Literature Review

2.1 Slurm and Kubernetes Integration Possibilities

Slurm and Kubernetes are distinct software environments that manage computational, storage, networking, and other resources and schedule computational work for various workloads. There is a strong and growing interest in AI/ML research. Many AI/ML applications are being developed in Docker containers with the expectation of deployment of these applications using Kubernetes. University research environments are becoming increasingly complex. University stakeholders and research computing directors should consider Kubernetes as a computational research environment to provide for their AI/ML researchers. Having an integrated environment and running both traditional HPC workloads with Slurm and cloud-native AI workloads with Kubernetes on the same resources should reduce costs compared to deploying additional cluster(s) for separate Kubernetes operation. Finding a flexible solution that allows resources to be on-premises, in the cloud, or in a hybrid configuration would be ideal. Finding a capability that can provide an integrated environment has been discussed by SchedMD's Chief Technology Officer (CTO), Tim Wickberg, in his SC'22 presentation "Slurm and/or/vs Kubernetes" (Wickberg (2024)) and his KubeCon 2025 presentation "Slinky: Slurm in Kubernetes, Performant AI and HPC Workload Management" (Wickberg (2025)) discussed an integrated implementation to be provided by SchedMD called the Slinky project (SchedMD (2025a,b)). Also, CoreWeave has mentioned on a blog related to KubeCon 2023 titled "Introducing SUNK: A Slurm on Kubernetes Implementation for HPC and Large Scale AI" (Coreweave (2023)) that they

have been working on the development of a Slurm and Kubernetes integration for their cloud environment that they call "Slurm on Kubernetes (SUNK)" (Coreweave (2023)) where "SUNK brings Kubernetes containerized deployments and Git Ops to Slurm and integrates a Slurm scheduler plugin to Kubernetes (Coreweave (2023))." Additionally, publications are showing strong interest in researching the integration of Slurm and Kubernetes with Unify "fusion scheduling with Slurm and Kubernetes" (Wu et al. (2024)) from researchers in China and "High-Performance Kubernetes" (Chazapis et al. (2024)) from researchers from Greece in the European Union.

SchedMD CTO Wickberg proposed in Wickberg (2024), from the Slurm perspective, three potential models that might be employed in a Slurm and Kubernetes integration: the "over" approach where Slurm is immortal and manages the cluster resources with Kubernetes clusters created from batch jobs and only available during the life of those job(s); the "adjacent" approach where Slurm and Kubernetes control planes exist on a converged cluster simultaneously with Slurm scheduling and prioritizing both Slurm jobs and Kubernetes Pod workloads with a proof-of-concept in development called slurm-k8s-bridge (SchedMD (2025f)); and the "under" approach where Kubernetes exists and a Slurm cluster is created within the Kubernetes environment with the Kubernetes workload outside of the view and control of Slurm. New features of dynamic nodes were added in Slurm version 22.05 in 2022, which provide the capability to provision nodes on the fly dynamically. Additionally, dynamic nodes can be dynamically added to network topologies in Slurm 25.05 (SchedMD (2025d)) in 2025. Both of these new features are necessary components for Slinky to provide close integration with Kubernetes and Slurm.

2.1.1 SchedMD Slinky

Wickberg (2024) and Wickberg (2025) describe the integration of Slurm and Kubernetes and the motivation and capabilities of SchedMD's Slinky. HPC workloads have traditionally run on clusters running Slurm. AI/ML training workloads can be run on Slurm or Kubernetes. AI inference applications usually run on Kubernetes. This complex set of workloads is converging in university research environments, and how they map to computational resources and the software systems that manage those resources are evolving and trending

towards convergence. The SchedMD Slinky project and its open-source software offer two possible integration capabilities for Slurm and Kubernetes. They are described on the SchedMD Slinky website SchedMD (2025a) and available on GitHub (SchedMD (2025b)). Slinky is fully supported and is in active development by SchedMD, the developers and supporters of Slurm. This fact makes Slinky a robust candidate for a pilot investigating the implementation of a Slurm and Kubernetes integration environment.

2.1.2 CoreWeave SUNK

Coreweave (2023) introduced CoreWeave's SUNK in October 2023. SUNK was developed to maximize computational resource utilization for AI training and inference workloads by integrating Slurm as a Kubernetes scheduler, enabling Slurm jobs to run within Kubernetes, and supporting both Slurm and Kubernetes workloads in the same environment. CoreWeave provides a wiki-like website with significant documentation on SUNK (Coreweave (2025)). Early versions of Coreweave (2023) communications indicated that CoreWeave would open-source SUNK in 2024. That indication was removed after June 2024, and a discussion with a CoreWeave sales team member in April 2025 indicated that SUNK was only available on the CoreWeave Kubernetes Service (CKS) and would not be made open-source. Since SUNK only runs in CoreWeave's CKS cloud environment and excludes an on-premises cluster, SUNK was not considered for the pilot described in Chapter 4.

2.1.3 Unify (China)

Wu et al. (2024) introduced the heterogeneous resource manager Unify as a fusion scheduling solution, combining Slurm and Kubernetes for partition deployment and/or hybrid deployment to meet the complex and diverse demands for computational resources. Problems addressed include dynamic node provisioning for the partition deployment scenario and resource scheduling conflict for a hybrid deployment scenario. Unfortunately, no reference to the availability of the Unify software is found in the paper references or on GitHub. Therefore, Unify was not considered for the Pilot described in Chapter 4.

2.1.4 High-Performance Kubernetes (European Union)

Chazapis et al. (2024) introduced High-Performance Kubernetes (HPK). HPK enables running unmodified cloud-native applications on an HPC cluster by allowing users to run user-level "mini Clouds" on a Slurm-managed HPC cluster. HPK utilizes a single container to run all Kubernetes control plane services. A Kubernetes Virtual Kubelet Provider, named hpk-kubelet, translates Kubernetes container management commands to a Slurm job with Apptainer for the container runtime. For HPK to work in the Slurm-managed HPC cluster environment, Apptainer must be configured to allow users to run containers with *–fakeroot* and must use a Kubernetes Container Network Interface (CNI) plug-in (for example, flannel CNI) for private IPs within containers. The goal of HPK is to enable HPC researchers to run Kubernetes workloads within a standard HPC cluster environment with Slurm. HPK delegates all configuration and scheduling to Slurm, maintaining Slurm's role of managing the cluster. HPK would be classified in the "over" class as described in Section 2.1 by transforming each Kubernetes deployment into a Slurm script. HPK is an open-source project, currently at version 0.1.2, and can be found on GitHub (FORTH ICS (2024)). The software was created and is maintained by developers at the Institute of Computer Science, Foundation for Research and Technology, Hellas, Greece, in the European Union. HPK is in its early stages of development, with its most recent release dating back to April 2024, and is maintained by a small team of developers. HPK was not chosen for the pilot since Open OnDemand and Apptainer provide a similar capability, albeit not supporting all Cloud-native workloads, and due to Slinky's more active development and support by SchedMD. HPK described in Chazapis et al. (2024) was built on prior works of Chazapis et al. (2023) and Zervas et al. (2022).

2.2 Apptainer

Many AI/ML applications are being developed in Docker containers. Many of these Docker containers can be converted and run on university HPC cluster environments using the Apptainer container platform. Apptainer is an open-source project, originally known as

Singularity, which was developed at the Lawrence Berkeley National Laboratory (Kurtzer et al. (2017)) and is discussed and documented at https://apptainer.org/ (Linux Foundation (2021)). Docker containers can be converted to the Apptainer *.sif* format and run by users on a Slurm-managed HPC cluster, either by running the Apptainer container on the login node or running the Apptainer container in a Slurm batch job or Slurm interactive batch job. The main limitation of Apptainer is that containers run by Apptainer can be run only by an unprivileged user. Consequently, Docker containers that rely on elevated root privileges for their operation cannot be converted to Apptainer format and run with these privileges using Apptainer. This is both a security feature for those managing a multi-user HPC cluster and a bug for researchers who rely on root privileges with Docker containers. Many AI/ML applications developed with Docker containers can leverage Apptainer on a university HPC cluster. However, AI inference is a class of AI/ML applications that can be very long-running and consume large amounts of computational resources upon demand, which does not fit the typical university HPC cluster resource and scheduling model, even when using Apptainer.

Chapter 3
Toward an Integration Solution

3.1 Slurm Overview

Figure 3.1: Slurm logo[1]

Slurm is a free and open-source workload manager for HPC clusters running Linux. Workload managers manage and allocate computational, storage, and other resources (i.e. software licenses) to user jobs usually maintained in a queue of prioritized jobs. Current versions of Slurm comprise a resource manager and scheduler and can be employed on small clusters up to the largest clusters. It has been estimated that more than half of the systems listed on the bi-annual top500.org list of the most powerful supercomputers use Slurm (Wikipedia (2025); USC (2019)). Slurm was initially developed at Lawrence Livermore National Laboratory (LLNL) in 2001 as a non-proprietary, open-source resource manager to support LLNL's desire to "move from proprietary high-performance computer systems to commodity hardware and open-source software" (SchedMD (2025e)). In 2010, a collaborative development effort started between LLNL, SchedMD (the company created to maintain and support Slurm), and three vendors to extend Slurm as a stand-alone resource manager and scheduler for any cluster. Slurm is available from various vendors that manufacture and sell HPC clusters, and is primarily used by universities, national laboratories, and other research institutions. The Slurm logo is shown in Figure 3.1.

[1] used by permission from SchedMD LLC. See Appendix A

Slurm was designed with the concept that the HPC cluster size is basically fixed, the workload is essentially infinite, and that the workload is comprised of interactive and batch jobs, which may include single-node and multi-node parallel work with time and resource limits that need some form of job queue scheduling prioritization (Wickberg (2024)). The Slurm training guide from SchedMD describes three key functions of Slurm:

> Slurm has three key functions. First, it allocates exclusive and/or non-exclusive access to resources (compute nodes) to users [jobs] for a specified period so they can perform work. Second, it provides a framework for starting, executing, and monitoring work (usually a parallel job) on the set of allocated nodes. Finally, it arbitrates contention for resources by managing a queue of pending work (SchedMD (2020a)).

Kubernetes design concepts are described in section 3.2 and it is important to understand the design differences between the two. Interest in Slurm has increased in the last five years in the public cloud, as shown by cloud vendors providing provisioning of Slurm clusters as part of their portfolio, including Microsoft Azure Cyclecloud (Microsoft (2025)), Lambda managed and unmanaged Slurm (Lambda (2025)), and CoreWeave SUNK (Coreweave (2025)).

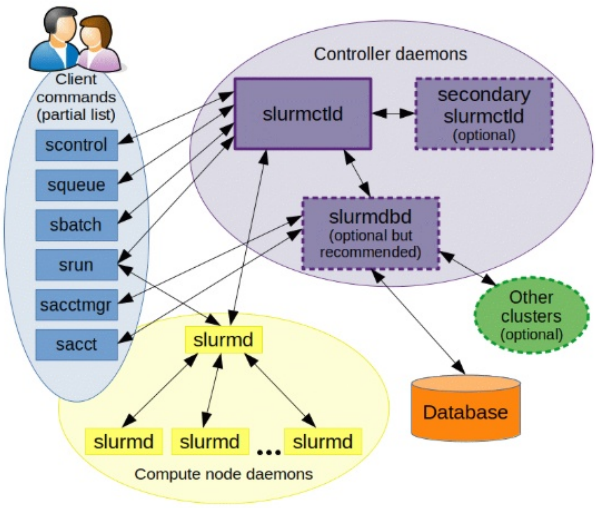

Figure 3.2: Slurm Architecture Overview[2]

[2]Image used by permission from SchedMD LLC. See Appendix A

Figure 3.2 shows the Slurm architecture overview. The architecture of Slurm consists of a central *slurmctld* master control daemon running on a cluster infrastructure node with an optional high availability fail-over duplicate running on another cluster infrastructure node, an optional - but highly recommended - *slurmdbd* database daemon running on a cluster infrastructure node for managing and archiving Slurm accounting records, and a *slurmd* daemon running on each compute node. The Slurm user and administrator client commands are usually run on a cluster login node and include, but are not limited to, the following commands: *sbatch*, *squeue*, *sinfo*, *sacct*, *salloc*, *scancel*, and *srun* for submitting a job to Slurm, getting job queue information, getting partition and other information, getting accounting information, setting up a resource allocation, canceling a job, and running a job's steps within an existing resource allocation (distinct from a batch job), respectively (SchedMD (2020b)).

The Slurm training guide describes succinctly the daemons (services) and the resources and tasks they manage:

> The entities managed by these Slurm daemons ... include nodes, the compute resource in Slurm partitions, which group nodes into logical sets (possibly overlapping), jobs, or allocations of resources assigned to a user for a specified amount of time, and job steps, which are sets of (possibly parallel) tasks within a job. The partitions can be considered job queues, each of which has a set of constraints, such as job size limits, job time limits, and users permitted to use it, among others. Priority-ordered jobs are allocated nodes within a partition until the resources (nodes, processors, memory, etc.) within that partition are exhausted. Once a job is assigned a set of nodes, the user can initiate parallel work in the form of job steps in any configuration within the allocation. For instance, a single job step may be started that utilizes all nodes allocated to the job, or several job steps may independently use a portion of the allocation (SchedMD (2020a)).

To interface with Slurm, users must create a batch job script and submit it using the *sbatch* command, or they can allocate resources with the *salloc* command and use them

interactively with the *srun* command. Both allocation and execution can be combined within a single *srun* command. For example, a user could create the below Slurm batch script shown in Listing 3.1 with the intended goal of printing out some information about the node that was allocated, printing out the resources available on the assigned node, and compile and execute a "hello world" C program. The contents of a Slurm job can be as simple as a Linux shell script, as complex as a message passing interface (MPI) parallel application, or other straightforward or complex constructs that may or may not be made up of a Docker container. If a Docker container is used, it must be combined with a tool that can execute the Docker container, such as Apptainer.

Listing 3.1: Sample Slurm batch script hello.sh

```
#!/bin/bash
#SBATCH -J hello              ## Name of job
#SBATCH -A isaac-utk0362      ## account to be charged
#SBATCH --partition=campus    ## partition to allocate nodes
#SBATCH --qos=campus          ## The quality of service
#SBATCH --time=00:01:00       ## Maximum time of the job
#SBATCH --nodes=1             ## Number of nodes
#SBATCH --ntasks-per-node=1   ## number of tasks/processors
#SBATCH -e hello.o%j          ## Errors will be written in this file
#SBATCH -o hello.o%j          ## output written here (same as above)
#########   Perform some simple commands   ######################
#########   Change to my Lustre scratch directory   #############
set -x
hostname
df -h -x tmpfx
cd /lustre/isaac24/scratch/$USER
#########   Compile and run the hello world application #########
gcc -o hello helloworld.c
./hello
```

Listing 3.2: Sample Slurm batch job submission

```
[vhazlewo@login1 vhazlewo]$ sbatch hello.sh
Submitted batch job 3826908
```

To submit the script to Slurm, one would use the command "*sbatch hello.sh*". Slurm will accept the batch job if it has no errors and return an assigned job number as shown in Listing 3.2. Once resources are allocated, the commands in the job script will be executed within a process on the assigned node. A file with all the output will be created in the file specified in the Slurm script directives section at the top of the job script, and the file will

be located in the directory that the user was in when the *sbatch* command was executed. Once the job is complete, the resources allocated to the user's job will be released by the Slurm scheduler for reallocation to new jobs.

Job status can be seen with the *squeue* command. Once the job is complete, then the output can be viewed in the file specified in the Slurm script directives using standard Linux commands, such as *cat*, shown in Listing 3.3.

Listing 3.3: Output created by running hello.sh Slurm batch script

```
$ cat hello.o3826908
+ hostname
ber1415
+ df -h -x tmpfx
Filesystem              Size  Used Avail Use% Mounted on
devtmpfs                504G     0  504G   0% /dev
tmpfs                   504G  1.5G  502G   1% /dev/shm
tmpfs                   504G  2.0G  502G   1% /run
/dev/mapper/xcatvg-root 442G   11G  431G   3% /
/dev/sda2               1014M  300M  715M  30% /boot
/dev/sda1               256M  5.9M  250M   3% /boot/efi
nfs:/data/nfs/home      13T   11T  1.6T  88% /nfs/home
nfs2:/data/nfs/sw       2.0T  1.6T  461G  78% /sw
nfs2:/data/nfs/spack    2.0T  1.6T  461G  78% /spack
/isaac24                4.5P  3.0P  1.3P  70% /lustre/isaac24
+ cd /lustre/isaac24/scratch/vhazlewo
+ gcc -o hello helloworld.c
+ ./hello
Hello World
```

Similarly, an interactive job can be performed by requesting the allocation of resources and specifying a shell with the *srun* command, as shown in Listing 3.4. Notice the prompt changed, indicating the change of the server resource the commands are running on, which is also shown by the hostname command output:

Listing 3.4: Command and output of a Slurm interactive job

```
[vhazlewo@login1 vhazlewo]$ srun -A isaac-utk0362 -N 1 -n 1 \
-p campus -q campus -t 1:00:00 --pty /bin/bash
[vhazlewo@sk131801 vhazlewo]$ cd /lustre/isaac24/scratch/vhazlewo
[vhazlewo@sk131801 vhazlewo]$ ls -l helloworld.c
-rw-r--r-- 1 vhazlewo tug2106 78 Jul 29 09:39 helloworld.c
[vhazlewo@sk131801 vhazlewo]$ hostname
sk131801
[vhazlewo@sk131801 vhazlewo]$ gcc -o hello helloworld.c
[vhazlewo@sk131801 vhazlewo]$ ./hello
```

```
Hello World
```

3.2 Kubernetes Overview

Figure 3.3: K8s logo[3]

Kubernetes, often referred to as K8s and pronounced "kates", was developed by Google to enhance the Google Cloud to compete with Amazon Web Services. In 2014, Google open-sourced and donated Kubernetes to the Cloud Native Computing Foundation (CNCF) (CNCF (2025a)). Kubernetes comes from the ancient Greek word κυβερνήτης for "helmsman" and the Kubernetes logo is a ship's helm wheel (shown in Figure 3.3) with seven handles.

Kubernetes was designed with the concept of a cloud-native system being essentially infinitely scalable, and the workload consisting of long-running microservices with little to no need for prioritization or multi-node work (Wickberg (2024)). A microservice is a small, independent application or service that has its own release cycles and can be deployed, scaled, and updated independently. Most cloud microservices are developed using portable, yet architecture-dependent, Docker containers. Once deployed to Kubernetes, these cloud-native applications can be auto-scaled, self-healed, auto-updated with rollout and/or rollback capabilities, and managed by Kubernetes services. These workloads are typically long-running. Examples include scalable enterprise web services (such as corporate websites and Google search), email servers (like Gmail), and now AI inference (Wickberg (2025)).

Nigel Poulton defined Kubernetes succinctly in The Kubernetes Book: "Kubernetes is an orchestrator of containerized cloud-native microservices applications" (Poulton and Joglekar (2025a)). Kubernetes is widely used in the public cloud. Amazon Web Services (AWS) provides Kubernetes with its Elastic Kubernetes Service (EKS). Google Cloud Platform (GCP) provides Kubernetes with its Google Kubernetes Engine (GKE). Microsoft Azure provides Kubernetes with its Azure Kubernetes Service. Oracle Cloud Infrastructure provides Kubernetes with its Oracle Kubernetes Engine (OKE). Kubernetes services can be extended, and the diagram at `https://landscape.cncf.io/` illustrates the current CNCF

[3] licensed use from CC-BY 4. See Appendix A

landscape for Kubernetes, which comprises five major categories: Application Definition and Development, Orchestration and Management, Runtime, Provisioning, and Observability and Analysis. Each of these has several subcategories and contains ten to hundreds of projects that can be integrated into Kubernetes, which are in various stages of deployment, with alpha, beta, and production being the most prominent stages. Kubernetes version 1.33 was released on 15 July 2025 (CNCF (2025c)).

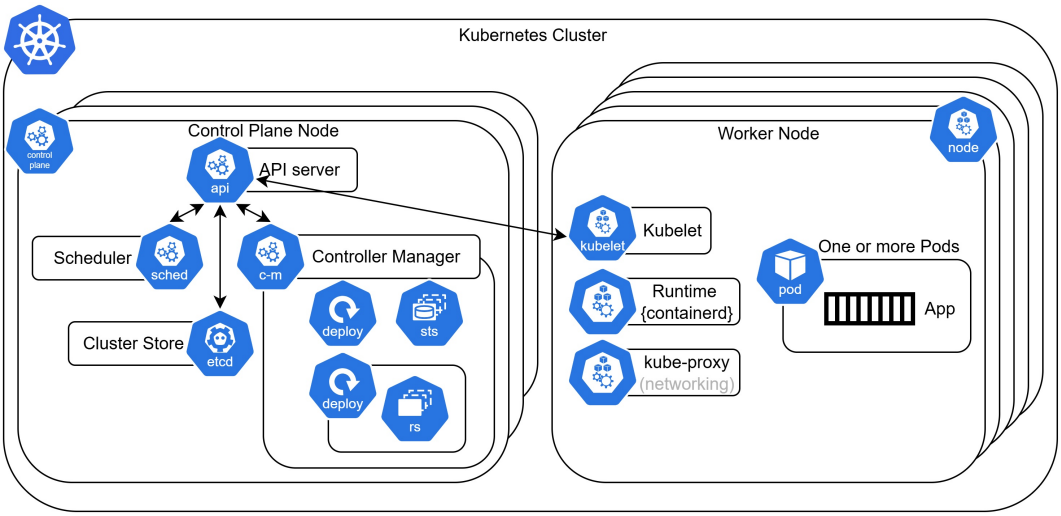

Figure 3.4: Example Kubernetes Cluster

At a high level, a cloud or data center comprises a complex collection of computational, storage, and networking resources. Kubernetes abstracts most of this away, making the resources easier to consume. Kubernetes is both a cluster and an orchestrator. A sample Kubernetes cluster is shown in Figure 3.4. Kubernetes has two node (server) types: Control Plane nodes and Worker nodes. Nodes may be virtual or physical servers, including instantiated cloud instances, or even containers, as is the case with using Docker Desktop to simulate a three-node cluster on a laptop. Control Plane nodes must be Linux-based, and worker nodes can be either Windows or Linux-based. A Control Plane is shown in Figure 3.4, and Control plane nodes run all the control plane services, including:

- **API server** - the Kubernetes interface
- **Scheduler** - monitors the API and schedules new tasks on resources

- **Controllers** - the services that automate the management and maintenance of resources (Controller Manager, Deployment, StatefulSet, ReplicaSet, Storage, etc.)
- **Cluster store (etcd)** - a database with the desired and current state information about the K8s cluster

A Kubernetes cluster can have as few as one Control Plane node, usually for testing, or as many as three to five or more for production environments, depending on the number of control plane services and high availability goals for a production environment. Large Kubernetes clusters with a high frequency of change should run the etcd Cluster Store on separate Control Plane nodes for better Cluster Store performance. A Worker environment is shown in Figure 3.4, and Worker nodes are where the microservices (applications) are run. The major components of a Worker node are:

- **Kubelet** - handles all the communication with the cluster, specifically, the api server
- **Runtimes** - software for executing containers on nodes, usually, *containerd* or *CRI-O*
- **Kube-proxy** - configures and maintains cluster networking and load balancer

Kubernetes can run containers, virtual machines, Wasm apps, and other services with all of these needing to be specified and wrapped in a Pod, where a Kubernetes Pod is a workload abstraction construct, before being handed off to Kubernetes and its underlying control services to orchestrate the complex tasks of choosing the appropriate node(s), joining networks, attaching storage volumes, maintaining state, scaling up or down, and other tasks on behalf of a microservice (application). These applications are usually single-purpose and require wrapping in a Pod to run on Kubernetes. Pods get wrapped in higher-level controllers (for example, a Deployments controller or a StatefulSet controller) for advanced features. Pods are usually stateless and immutable, which is another difference between Kubernetes and Slurm. There are single-container Pods and multi-container Pods, with multi-container Pods implementing an application with a complementary service or functionality. Typically, this is achieved by running the application in a helper container, known as a Service Mesh sidecar, rather than running the same application container multiple times within a Pod. A single container Pod and a multi-container Pod are shown in 3.5. Multi-container Pods

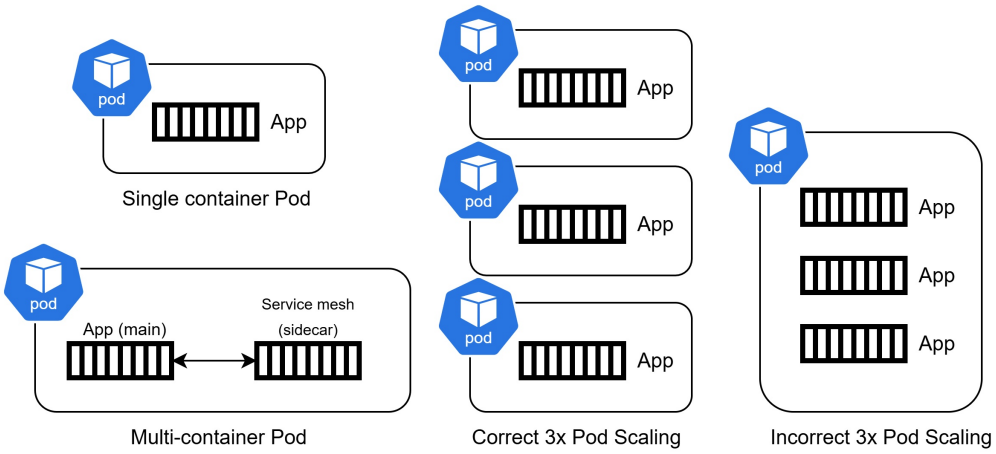

Figure 3.5: Pod Examples

further the implementation concept that every container should perform a single task. Each Pod is a shared execution environment for one or more containers. A single-container Pod has exclusive access to the execution environment, while containers in a multi-container Pod share the same execution environment. This execution environment includes a network stack, file system, storage volumes, shared memory, hostname, resource limits, and other components. Kubernetes schedules Pods, not containers, and Pods are essentially the atomic unit of scheduling in Kubernetes. Scaling an application up involves adding more Pods, while scaling it down is achieved by deleting Pods. Scaling an application is performed by adding more replicant Pods not by adding more containers (with the same application) to an existing Pod. Scaling an application is shown in Figure 3.5.

Pods are lightweight, add minimal overhead, abstract workload details, and can be composed of Docker containers, VMs, Wasm apps, and other applications. Since it is an abstraction, Kubernetes does not evaluate the application contents of a Pod. Kubernetes extends Pods by wrapping them with deployment controllers that have capabilities and features beyond the workloads they contain, including, but not limited to, resource sharing, advanced scheduling (such as scaling, upgrade rollout, version rollback, etc.), application states, termination control, storage volumes, and restart and security policies. These controllers include, but are not limited to, Deployments, ReplicaSets, StatefulSets, and DaemonSets. Figure 3.6 shows three Pod deploy and manage scenarios.

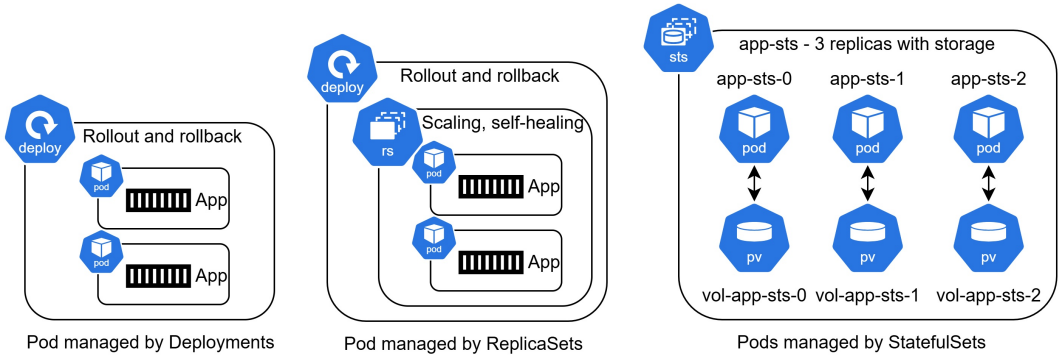

Figure 3.6: Pod Deploy and Manage Examples

Every Kubernetes cluster implements a Pod network via the Container Network Interface (CNI), which is typically a Layer 2 network that spans all cluster nodes. This Pod network enables every Pod to communicate with all other Pods, regardless of the node on which the Pod is located. Figure 3.7 shows six Pods running on three nodes and how the Pod network spans all three nodes and is separate from the node network. Nodes have names and optionally labels. Pods can be assigned labels, and thereby the Kubernetes scheduler can be instructed to schedule labeled Pods onto nodes with corresponding labels, ensuring that Pods with specific labels run only on nodes with those labels.

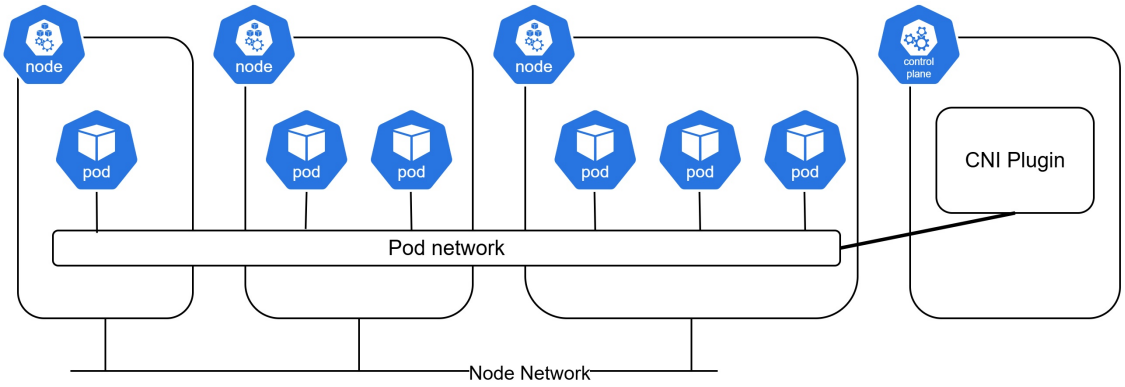

Figure 3.7: Pod Network

Kubernetes also has the concept of Namespaces, which can be used to divide a Kubernetes cluster for use by different applications, users, projects, and/or departments of an organization. This can be very useful in the context of implementing a Kubernetes cluster in a university campus environment, similar to what is done with Slurm clusters, to provide isolation of applications, users, and/or groups within the same cluster, with weaker

isolation than the strong isolation and cost of providing separate clusters for the isolation. Namespace examples are:

- dev - the development namespace
- prod - the production namespace
- condo-A - the private resource namespace for group A

or

- campus - the namespace for use by any campus participant
- dept-cs - the namespace for use by participants from the Computer Science department
- dept-business - the namespace for use by participants from the Business department

In its simplest form, users interface with Kubernetes with the *kubectl* command to submit, monitor, and manage work with Pods. The definition of a Kubernetes Pod could be created in a YAML file prepared with a file editor, such as vi. For example, the YAML file mypod.yaml would contain the Pod definition information for the application to be run on the Kubernetes cluster. An example mypod.yaml file is shown in Listing 3.5 that would create a Pod from a Docker container defined in the specified Docker container image that would run a "*hello world*" style webserver on port 80.

Listing 3.5: Sample Kubernetes yaml file declaring a single webserver in a Pod

```
apiVersion: v1
kind: Pod
metadata:
  name: hello-webserver
  labels:
    version: v1
spec:
  containers:
  - name: hello-container
    image: victorhazlewood/webserver:1.0
    ports:
    - containerPort: 80
    resources:
      limits:
        memory: 128Mi
        cpu: 0.5
```

That YAML file would then be presented to the Kubernetes API server with the command "*kubectl apply -f mypod.yaml*". The Pod could be monitored with this simple command "*kubectl get pods*". Requesting all the information Kubernetes knows about the Pod from the yaml file and any defaults could be with this command "*kubectl get pods hello-webserver -o yaml*". This command can provide an overview of the pod state and any event status "*kubectl describe pod hello-webserver*". The web server would run on the Kubernetes cluster until the Pod was deleted, which could be done with the following commands "*kubectl delete pod hello-webserver*" or "*kubectl delete -f mypod.yaml.*" Of course, there are graphical user interfaces, such as Rancher (Rancher Labs (2020)), that provide a browser-based interface to Kubernetes for users and administrators, rather than a command-line interface. Such tools are beyond the scope of this thesis, but they would be an important element for enhancing Kubernetes usability within the research community.

Notably, Kubernetes can be extended with a Kubernetes Operator (CNCF (2025b)), which is essentially a Kubernetes Pod Controller that runs on a worker node rather than on the control plane node(s). A Kubernetes Operator is a specification of packaging, deploying, and managing a Kubernetes application that extends Kubernetes by linking a special application-specific Controller to one or more Custom Resources. This Controller for these Custom Resources runs outside of the Control Plane, like any other containerized application, and is a client of the Kubernetes API. This enables the automation and lifecycle management of application-specific tasks and services within a Kubernetes cluster.

Kubernetes architecture and operation are significantly different than Slurm. This difference will require university research computing staff who are not familiar with Kubernetes, including system administrators, scientific computing support, student workers and management, to obtain training, knowledge, and experience in installing, operating, maintaining, managing, and supporting Kubernetes for research computing purposes. Of specific importance would be the allocation and usage of compute and storage resources for AI inference, which is likely a new research computing application modality.

3.3 Slurm and Kubernetes Integration with Slinky

Slurm and Kubernetes have very different design concepts including: the view of the workload having access to a fully-featured Linux environment (Slurm) or packaging up the complete environment needed for the application as part of the application deployment (Kubernetes), the view of the scalability of the computational resources available for the workload Slurm (finite) or Kubernetes (infinite) manages, and their view of the resource limits and prioritization (Slurm) or lack of the use of prioritization (Kubernetes) of the workload being managed (Wickberg (2024, 2025)). The centralized, general-purpose university research HPC computing environments have an installed base of Slurm-based HPC clusters; yet, a significant portion of AI research projects and tools are based on Docker containers and have some expectation of deployment with Kubernetes (i.e., NVIDIA NIM, NVIDIA (2024a) and Hugging Face Hub Spaces, Raina et al. (2025)). Additionally, setting up separate HPC clusters that utilize Kubernetes, distinct from the existing Slurm-based HPC clusters, would incur an additional expense for universities to bear and support. A review of university research computing websites from section 1.3 indicates many university HPC clusters utilize Apptainer to bridge the gap. However, there are still issues to address, especially with AI inference applications, which may map better to Kubernetes environments than Slurm environments due to their ease of installation, use, maintenance, and the efficient consumption of computational and storage resources by long-running AI inference workloads. AI inference workloads may be finite when in testing and development, but then become infinite (or very long-running, thereby locking up resources) when used in production. AI inference for testing and development can be considered AI inference development, while AI inference for production can be regarded as enterprise AI inference.

Slinky from SchedMD, mentioned in section 2.1.1 and described in section 3.4, could provide a solution that integrates the capabilities of Slurm and Kubernetes while being deployable on a single HPC cluster environment. This could enable the use of Kubernetes and Slurm for researchers who require the capabilities provided by either, eliminating the need to deploy a separate HPC cluster. As described by SchedMD's CTO Tim Wickberg in his 2023 presentation, convergence of HPC and cloud-native approaches could benefit

both ecosystems from each other: "Kubernetes [could benefit] from increased throughput, different approaches to job scheduling and prioritization" and "Slurm [could benefit] from newer cloud native technologies and tools, and increased focus on flexibility in support of new user workflows (Wickberg (2024))."

3.4 Slinky Overview

Figure 3.8: Slinky logo[4]

SchedMD created Slinky as HPC systems are increasingly being requested to be used for AI/ML workloads, as batch-style AI/ML workloads are being requested on Kubernetes systems (for example, AI training) and the apparent conclusion is that running HPC and cloud-native workloads on separate clusters in the university environment can be inefficient and costly. Slinky has two separate projects that can be deployed to integrate Slurm and Kubernetes, tailored to specific use cases: slurm-operator, which deploys a complete Slurm cluster operating within Kubernetes, and slurm-bridge, which enables the co-location of Kubernetes and Slurm on a set of cluster resources. These capabilities have different use cases for integrating Kubernetes and Slurm. Still, they both allow use of many of the workload and resource management capabilities of Slurm, such as, scheduling priority and preemption, workload policies and resource limits, MPI job support, and accounting.

The Slinky slurm-bridge augments a Kubernetes environment to run Slurm as a Kubernetes scheduler for non-critical Pods created using specific namespaces. The slurm-bridge contains a Slurm Bridge Admission Controller, which is a webhook for Pods created in specific namespaces, that modifies these Pods to use the Slurm Bridge Scheduler by modifying the Pods .spec.schedulerName, and a Slurm placeholder job is created to represent the Pod workload (SchedMD (2025f)). The Kubernetes kubelet and the Slurm slurmd are co-located on the same physical host with the Slurm Bridge Scheduler being in complete control of the workload bound to those managed nodes. The slurm-bridge managed Pod is scheduled by

[4]used by permission from SchedMD LLC. See Appendix A
[5]used by permission from SchedMD LLC. See Appendix A

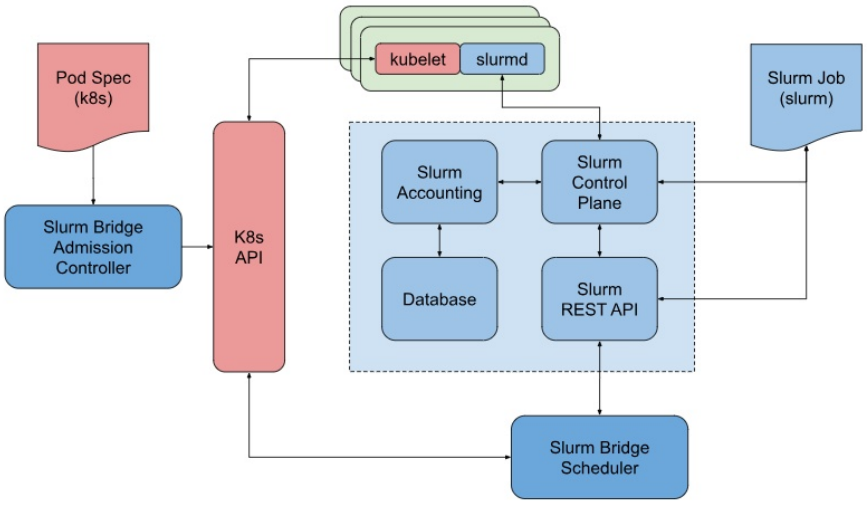

Figure 3.9: slurm-bridge Architecture Overview[5]

the Slurm Bridge Scheduler, which informs Kubernetes of the node to allocate for the Pod. The kubelet then launches the Pod instead of slurmd. Note that this Slinky slurm-bridge functionality has the limitation that the Kubernetes Pod will receive an exclusive, whole node allocation for the Pod. The slurm-bridge architecture is shown in Figure 3.9.

The Slinky slurm-operator is a Kubernetes Operator (discussed at the end of Section 3.2) that is deployed and manages the components of a Slurm cluster within a Kubernetes cloud-native environment. The slurm-operator enables both workload managers, unified on Kubernetes. The slurm-operator extends the Kubernetes API and acts as a Kubernetes controller for a set of Custom Resources, including some resources that could be external to Slurm and Kubernetes, such as LDAP. Using slurm-operator, a Kubernetes cluster is created, and then all the services of a Slurm cluster are created within this Kubernetes cluster as Pods, Services, or Custom Resources, allowing users to run workloads created and managed by Slurm on a Kubernetes cluster. The slurm-operator architecture is shown in Figure 3.10. Also, the Slinky slurm-operator allows for hybrid configurations where some components of the Slurm configuration are external to the Slurm cluster contained within Kubernetes and could be services that are located on bare-metal or VMs outside of Kubernetes, such as LDAP, network file system, a MariaDB database, or compute/GPU nodes. An example hybrid configuration is shown in Figure 3.11.

Due to the flexibility of Slinky slurm-operator in configuration of the integration resources, the ability to have some Slurm services be hybrid and external to the Slurm environment contained within Kubernetes, and the ability to run native HPC workloads and native Kubernetes workloads, the Slinky slurm-operator was chosen for the integration pilot described in Chapter 4.

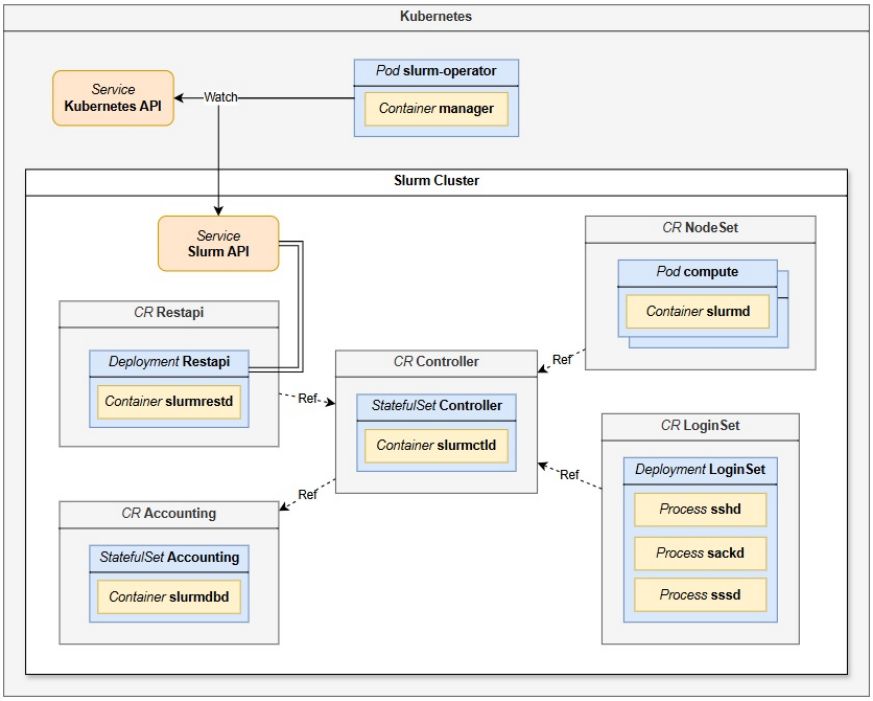

Figure 3.10: slurm-operator Architecture Overview[6]

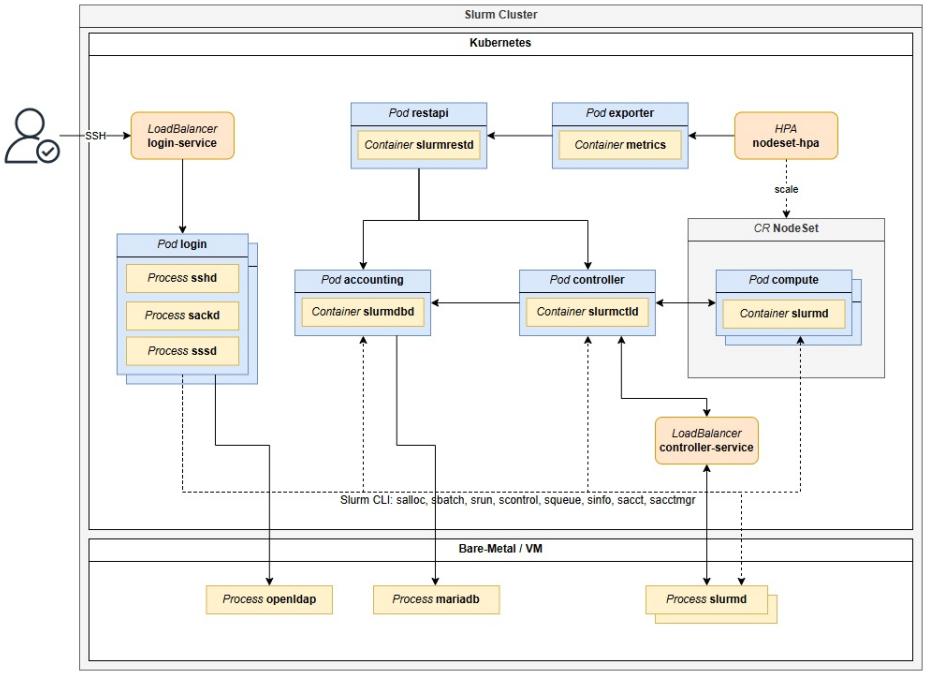

Figure 3.11: Slinky slurm-operator Hybrid Configuration[7]

[6]Image used by permission from SchedMD LLC. See Appendix A
[7]Image used by permission from SchedMD LLC. See Appendix A

Chapter 4
Slurm and Kubernetes Integration Pilot

4.1 ISAAC Next Generation Cluster

UTK provides an HPC cluster environment to support the computational research activities of the flagship UTK campus and all University of Tennessee campuses, in alignment with their academic and research missions. This environment is managed by the High Performance & Scientific Computing (HPSC) group of the Office of Innovative Technology, the information technology division of UTK. The flagship HPC cluster is the Infrastructure for Scientific Applications and Advanced Computing (ISAAC) Next Generation (ISAAC NG) HPC cluster, which is housed in the Kingston Pike Building data center on the UTK campus in Knoxville, TN. Photos of ISAAC NG are shown in Figure 4.1. As of July 2025, ISAAC NG consisted of a collection of infrastructure servers, 216 conventional compute servers, 33 GPU servers with 91 GPUs, and a 5 petabyte formatted (6.5 petabyte raw) Lustre file system built on a Data Direct Networks ES400NVX storage subsystem with an HDR InfiniBand interconnect. The computational servers that make up the compute and GPU servers are described in Table 4.1. For the rest of this document, Kubernetes is abbreviated as "K8s".

4.2 Slinky Pilot Environment

A subset of compute resources of the ISAAC NG cluster was provided by the HPSC group for the Slinky pilot environment. The resources provided included three Skylake compute servers, one anticipated for the K8s Control Plane (node sk131601) and two as K8s Worker

Table 4.1: ISAAC NG Compute and GPU servers

#	Server Type	Processor *	Cores	Memory	GPU **
8	Dell R640	Skylake	40	192 GB	
1	Dell R740	Cascade Lake	40	384 GB	4xV100
122	Dell R640	Cascade Lake R	48	192 GB	
4	Dell R640	Cascade Lake R	48	1.5 TB	
1	Dell R740	Cascade Lake R	48	192 GB	4xT4
1	Dell R740	Cascade Lake R	48	192 GB	V100S
6	Dell R740	Cascade Lake R	48	192 GB	2xV100S
9	Dell R740	Cascade Lake R	48	768 GB	2xV100S
38	Dell R650	Ice Lake	56	256 GB	
3	Dell R650	Ice Lake	56	2 TB	
3	Dell R650	Ice Lake	56	256 GB	2xA16
2	Dell R650	Ice Lake 8358	64	256 GB	
1	Dell R650	Ice Lake 8358	64	2 TB	4xA40
5	Dell R660	Sapphire Rapids	64	512 GB	
1	Dell R760xa	Sapphire Rapids	64	768 GB	2xL40
10	Dell XE8640	Sapphire Rapids	64	768 GB	4xH100
17	Dell R7625	AMD Bergamo	224	1 TB	
1	Dell R6515	AMD Rome	32	128 GB	
1	Dell R6625	AMD Genoa	64	1 TB	
1	Dell R6625	AMD Genoa	128	512 GB	
2	Dell R6625	AMD Genoa	128	768 GB	
4	Dell R6625	AMD Genoa	128	1.5 TB	
1	Dell R6515	AMD Milan	32	128 GB	
7	Dell R6515	AMD Milan	128	512 GB	

* - Intel processors unless indicated; ** - NVIDIA GPUs

Figure 4.1: ISAAC NG racks front, rear, and server views

nodes (nodes sk131802 and sk131804), and a Dell R740 GPU server (clv1530) with four NVIDIA V100 GPUs as the third K8s Worker node. Storage was anticipated to come from the storage on these servers or from the Lustre storage, but as described in the next section, this was an incorrect assumption and became problematic.

4.3 Pilot Installation, Testing, and Results

K8s version 1.32.6 was installed using the four nodes described in section 4.2. The output from the "*kubectl get nodes -A*" command showing the pilot K8s node information is shown in Listing 4.1. Note that only sk131601 has a defined K8s role. A list of the K8s Pod information of the controllers and services running on the control-plane is shown in Listing 4.2. A calico-node and kube-proxy Pod is running on each configured K8s node, which is not shown. Five basic Pod examples from *The Kubernetes Book* were run successfully and then deleted, which demonstrated that K8s was working.

Listing 4.1: Command and output to show pilot node information

```
# kubectl get nodes -A
NAME        STATUS   ROLES           AGE   VERSION
clv1530     Ready    <none>          3h    v1.32.6
sk131601    Ready    control-plane   15d   v1.32.6
sk131802    Ready    <none>          15d   v1.32.6
sk131804    Ready    <none>          15d   v1.32.6
```

Listing 4.2: K8s Pod information

```
Namespace      Node      State    Name
kube-system  sk131601  Running  calico-kube-controllers-c47bd57c5-zddj8
kube-system  sk131601  Running  coredns-668d6bf9bc-kw64n
kube-system  sk131601  Running  coredns-668d6bf9bc-qshjd
kube-system  sk131601  Running  etcd-sk131601
kube-system  sk131601  Running  kube-apiserver-sk131601
kube-system  sk131601  Running  kube-controller-manager-sk131601
kube-system  sk131601  Running  kube-scheduler-sk131601
```

When the pilot work started, Slinky slurm-operator version 0.3.1 was the latest version available from GitHub (SchedMD (2025b)). The slurm-operator Quickstart Guide instructions for Slinky slurm-operator version 0.3.1 (SchedMD (2025c)) did mention the software requirements for installing cert-manager, a K8s certificate management system, and

Prometheus, a K8s metrics, monitoring, and alerting system. However, the instructions did not provide information on how much server resources should be allocated for Prometheus, cert-manager, or Slinky slurm-operator Pods (services). Also, no storage requirements were specified. Additionally, with K8s capabilities for namespaces and labels to tie the affinity of Pods (services) to specific resources, there was no mention or guidance on how to use K8s namespaces or labels to appropriately home these Pods or services to specific server resources. Server and storage minimum resource specification would be of interest to system administrators for production operation purposes. Therefore, issue #52 was created against the GitHub Slinky Project slurm-operator repository (Hazlewood (2025)) regarding storage requirements and service to node affinity instructions. Recall that Figure 3.10 shows the Pods and Custom Resources (CR) that make up the Slinky slurm-operator. Listing 4.3 shows the Pods installed following the Slinky slurm-operator QuickStart Guide for the software from the provided Helm charts.

Listing 4.3: Slinky Pod v0.3.1 information

```
Namespace     Node      State     Name
cert-manager  clv1530   Running   cert-manager-58dd99f969-7fkxt
cert-manager  clv1530   Running   cert-manager-cainjector-55cd9f77b5-x4swx
cert-manager  sk131804  Running   cert-manager-webhook-7987476d56-9h7qm
prometheus    sk131804  Running   alertmanager-prometheus-kube-prometheus-alertmanager-0
prometheus    sk131804  Running   prometheus-grafana-5d8f8df89f-zd997
prometheus    clv1530   Running   prometheus-kube-prometheus-operator-9945dc645-6hv22
prometheus    sk131802  Running   prometheus-kube-state-metrics-75dff79b9f-p8sjx
prometheus    sk131802  Running   prometheus-prometheus-kube-prometheus-prometheus-0
prometheus    sk131601  Waiting   prometheus-prometheus-node-exporter-gtrxn
prometheus    clv1530   Waiting   prometheus-prometheus-node-exporter-j4gv2
prometheus    sk131804  Waiting   prometheus-prometheus-node-exporter-knrjs
prometheus    sk131802  Waiting   prometheus-prometheus-node-exporter-lr8t8
slinky        clv1530   Running   slurm-operator-64f5dddf8d-fghfs
slinky        clv1530   Running   slurm-operator-webhook-54bcd65466-n44sl
slurm         clv1530   Running   slurm-accounting-0
slurm         clv1530   Waiting   slurm-exporter-84cd7d7769-s7kcn
slurm         clv1530   Running   slurm-login-85d748bc87-nslrp
slurm         clv1530   Running   slurm-restapi-558d5b57d4-x92hz
slurm         clv1530   Running   slurm-token-create-79s22
slurm         <none>    FailnoPVC slurm-mariadb-0
              FailedScheduling
              0/5 nodes are available:
              pod has unbound immediate PersistentVolumeClaims
slurm         <none>    FailnoPVC slurm-controller-0
              FailedScheduling
              0/5 nodes are available:
              pod has unbound immediate PersistentVolumeClaims
```

Upon initial installation of Slinky slurm-operator version 0.3.1, it was noted that two Pods failed, the slurm controller and the slurm mariadb database, both of which are necessary for successful operation of the Slinky slurm-operator. These failed due to no storage definition, specifically K8s StorageClass and PersistentVolume definitions, related to the Pod requests

for a PersistentVolumeClaim (PVC). During the time it took to investigate the failed Pods for the Slinky slurm-operator, Slinky slurm-operator version 0.4.0 was announced.

Slinky slurm-operator version 0.3.1 was deleted (K8s term for uninstalling) in anticipation of installing version 0.4.0. It was also noted that the Slinky slurm-operator developers stated that the transition to new versions of the form 0.#.0 will not be backward compatible (Hafener (2025)). To address the desire to have unused Worker nodes available for K8s and Slurm jobs and to control Slinky slurm-operator (service) Pod placement for cert-manager, Prometheus (Prometheus has four services and also runs on each node in the K8s cluster shown in Listing 4.3), slinky, and slurm Pods, the Worker nodes sk131801 and clrv1530 were removed from K8s, and then Slinky slurm-operator version 0.4.0 was installed. Prometheus was not installed due to not needing metrics or monitoring until it could be demonstrated that Slinky slurm-operator could be installed and made operational.

Listing 4.4: Slinky Pod v0.4.0 information

```
Namespace     Node      State    Name
kube-system   sk131601 Running calico-kube-controllers-c47bd57c5-zddj8
kube-system   sk131601 Running coredns-668d6bf9bc-kw64n
kube-system   sk131601 Running coredns-668d6bf9bc-qshjd
kube-system   sk131601 Running etcd-sk131601
kube-system   sk131601 Running kube-apiserver-sk131601
kube-system   sk131601 Running kube-controller-manager-sk131601
kube-system   sk131601 Running kube-scheduler-sk131601
kube-system   sk131601 Running calico-node-864gd
kube-system   sk131802 Running calico-node-px4t4
kube-system   sk131804 Running calico-node-qc74w
kube-system   sk131601 Running kube-proxy-pwmx5
kube-system   sk131802 Running kube-proxy-4xn4d
kube-system   sk131804 Running kube-proxy-9hl5t
cert-manager sk131802 Running cert-manager-7ccbc46c67-9l72f
cert-manager sk131802 Running cert-manager-cainjector-655bb66698-z85z9
cert-manager sk131802 Running cert-manager-webhook-586666dcd7-jjtvx
mariadb       sk131804 Running mariadb-operator-5d6678d898-9qkg5
mariadb       sk131802 Running mariadb-operator-cert-controller-8f65988b9-d8d9f
mariadb       sk131804 Running mariadb-operator-webhook-56fbc45fb9-jk5dp
slinky        sk131802 Running slurm-operator-558bbd798c-q6vdl
slinky        sk131802 Running slurm-operator-webhook-6574f47f78-kwjd2
slurm         sk131802 Running slurm-accounting-0
slurm         sk131802 Running slurm-exporter-7d4f79bb85-g6ghr
slurm         sk131802 Running slurm-restapi-6c67d5c98-ncxn8
slurm         sk131802 Running slurm-worker-slinky-0
slurm         <none>   Pending mariadb-0
              FailedScheduling
              0/3 nodes are available:
              pod has unbound immediate PersistentVolumeClaims
slurm         <none>   Pending slurm-controller-0
              FailedScheduling
              0/3 nodes are available:
              pod has unbound immediate PersistentVolumeClaims
```

Listing 4.4 shows the new list of Pods, their node placement, and the same PVC storage failure. The listing shows all the K8s services running on the control plane node, except

those services that should run on each node (calico-node and kube-proxy). It also shows the cert-manager, slinky, and slurm namespace services running on sk131802, the mariadb-operator services running on sk131802 and sk131804, and unassigned nodes for the two failed services, which failed due to no storage defined to satisfy the PVC.

To satisfy the storage PVC requirements for MariaDB and the Slurm controller specifically, and more generally for the K8s environment, a selection from one of the Container Storage Interface (CSI) drivers (GitHub K8s CSI (2025)) would need to be identified and installed in the K8s environment. The list of choices is available from the K8s CSI Drivers website (GitHub K8s CSI (2025)). Available storage options could include an unused solid-state device (SSD) configured and defined on one or more K8s nodes for use by the Rook CSI driver, or the Lustre Exascaler storage system could be made available on ISAAC NG with an Exascaler CSI driver. Rook requires at least one empty and completely unused SSD, and an unused SSD does not exist on any of the pilot resources. The use of SSDs with Rook is a suitable choice for testing, but would not be preferred for production use on the ISAAC NG cluster. For production use, Lustre would be preferred. An investigation of the Lustre Exascaler CSI Driver on GitHub from Data Direct Networks revealed that the latest release, version 2.4.0 from 23 June 2025, requires an Exascaler client version 2.14.0 or higher and an Exascaler server version 6.3.2 or higher. Any Exascaler CSI driver of version 2.3.0 or above has this requirement. The current client on ISAAC NG is 2.12.9 and the current Exascaler server is 6.3.1. To use a current Exascaler CSI driver the ISAAC NG Lustre client and Exascaler server would both have to be upgraded. To upgrade Lustre clients is relatively straightforward, but upgrading the Lustre server is a very complex operation that would require significant planning and scheduling of the upgrade. The upgrade would require making the /lustre/isaac24 file system unavailable and idling the entire ISAAC NG cluster. The upgrade was not performed as part of the pilot primarily due to time constraints.

4.4 Slinky slurm-operator Github Issues

The Slinky slurm-operator is supported by SchedMD, available on GitHub, and users of the Slinky slurm-operator are encouraged to report any issues. Reviewing the Slinky slurm-operator issues and the subsequent comments can give valuable insight into problems people are encountering, information about the availability of releases and features, and other helpful information. By September 26th, there were twenty reported Slinky slurm-operator issues, with five open and fifteen closed. Table 4.2 shows the type and number of issues reported in GitHub for Slinky slurm-operator. Although the sample size is small, the number of feature requests is notable, as it is close to the number of software bugs reported. Additionally, helpful information was discovered by reviewing all the reported issues, as shown in Table 4.3.

Table 4.2: Slinky slurm-operator issues from GitHub

Type	Number
Software bugs	9
Installation	3
Documentation	1
Feature Requests	7

Table 4.3: Helpful information from GitHub issues

Guide to upgrade information (#40)
Pointer to guide for using containers in Slurm (#41)
Reference to Pyxis and Enroot use (#26)
Commands used to troubleshoot Pod issues (#44)
Sample helm values files used for installing slurm-operator (#44, #45)

4.5 Pilot End and Lessons Learned

To be viable for production, K8s and Slinky slurm-operator need reliable and ample storage. There were no available SSDs for Rook storage for the pilot and Lustre version mismatches prevented use of the existing ISAAC NG Lustre for the pilot. The pilot was ended due to these storage issues and noting that the Slinky slurm-operator upgrades to new versions

would be incompatible with previous versions requiring upgrades to be full uninstalls followed by full installations. Lesson learned from the pilot included:

- the need for addressing storage as part of any initial K8s installation,
- the installation of K8s, even without Slinky, was very useful to expose and familiarize university support staff to the environment and the potential to extend the portfolio of research capabilities,
- identifying the need to address the complexities of K8s production administration and usability with a tool, such as, Rancher,
- being aware that Slinky slurm-operator version upgrades would require full uninstall of an old version and complete install of new version along with addressing any new or changed helm chart values that may arise with new versions,
- the need for addressing the K8s and Slinky slurm-operator service node affinity issues to make sure there are unused and unallocated compute and GPU resources for user Pods,
- and the need for a list of the most desired features and capabilities, as well as, using the list to test future Slinky versions for these features and capabilities.

A review of the GitHub documentation and issues described for Slinky slurm-operator 0.3.1 and 0.4.0 gave indications that Slinky slurm-operator has much promise as a system that could share and manage a single hardware resource environment but is not yet ready for deployment in the UTK production HPC environment.

Chapter 5
Conclusions and Future Work

5.1 Conclusions

The landscape of information technology, high-performance computing, and university research computing is constantly changing. Slurm is the de facto standard for resource and workload management in various academic and research environments with university high-performance computing clusters. K8s is a cloud-native, open-source software system for managing resources and workloads of containerized applications. It is considered the de facto standard for AI applications and orchestrated workflows. Conventional and accelerated computational resources and storage resources are widely available in university HPC clusters, along with Slurm for managing and scheduling jobs on those clusters. K8s services are typically absent from university campus HPC clusters.

There is increasing interest among university researchers in developing and using AI applications using university HPC cluster resources. With this growth in interest in AI research, it has gained significant momentum, presenting some distinct computational requirements that differ from traditional use cases for an HPC cluster using Slurm. With the prevalence of AI applications being developed in Docker containers intended to be deployed on the widely adopted K8s platform and with the allocation of resources for low-latency response times for AI inference use cases, it is concluded that there is a need to deploy K8s to the portfolio of services available on university HPC clusters to facilitate AI workloads.

Depending on individual university research computing environment goals and budgets, K8s could be installed and maintained on a separate HPC cluster resource. This choice would

likely add additional cost to a university research computing budget and staff which may already be strained. Alternatively, integrating K8s and Slurm onto the same HPC cluster resource is also possible with one of the solutions outlined in Chapter 2, which could reduce hardware and other costs. The base components of K8s are open-source and free for use. The components of SchedMD's Slinky are open-source and free for use, with training and support available from SchedMD for a fee. Slinky provides integration and interoperability capabilities to enable the co-location of K8s and Slurm on the same HPC cluster resources with either slurm-bridge or slurm-operator. As described in Chapter 3, K8s is quite different from Slurm. To support AI inference workloads and other K8s workloads, K8s should be added to the research computing environment portfolio. K8s can be deployed on a separate HPC cluster or by integrating K8s and Slurm onto the same HPC cluster with Slinky, with the decision likely to be based on costs, funds availability, staffing, and features and capabilities of Slinky. In addition, if K8s is added to the portfolio, the university research computing staff will need training and other support with K8s to gain the knowledge and experience in installing, maintaining, and supporting a K8s environment, along with similar training for Slinky. For typical university research computing environments, such as those in the SEC listed in Table 1.1, this work concludes that adding K8s and Slinky slurm-operator to the research computing environment portfolios of universities with active AI research programs is a near-term viable solution for adding K8s to the research computing portfolio while also attempting to reduce costs by using the same hardware available in the existing HPC cluster with Slurm. Slinky slurm-operator is in active development by a proven vendor that supports the goals of the open-source and research communities and shows promise as a solution to integrate K8s and Slurm into a single resource environment, along with extensions to allow for flexible configurations. As shown in the Slinky slurm-operator hybrid configuration in Figure 3.11, some existing infrastructure components of the existing Slurm environment could be reused (i.e. LDAP), or a separate Slurm environment could be deployed, as shown in Figure 3.10. By deploying K8s with Slinky slurm-operator to a university research computing portfolio, it can address the current and future needs of AI researchers, either with on premises resources or in a public cloud.

5.2 Future Work

Some additional items beyond the scope of this thesis that should be investigated as part of a comprehensive review of items for installation and use of Slinky in a university environment includes the following:

- For UTK specifically, upgrade of Lustre clients and servers to versions compatible with the latest version of Lustre Exascaler CSI driver for K8s should be addressed as part of the prerequisite infrastructure required for installation of a K8s environment. Slinky slurm-operator issue #52 was submitted requesting to address storage and server requirements in the prerequisites in the installation guide (for version 0.4.0 and beyond). This will allow the Lustre storage already available to be used with the K8s environment with or without Slinky slurm-operator installation.

- Integration of K8s with an administrator and user interface, such as, Rancher (Rancher by SUSE (2025)), should be considered as part of any K8s environment to address centralized management, administrative capabilities, and authentication, authorization, and access control through a web-based user interface.

- The cost of training, installation, and maintenance support to the existing system administrators and support staff to address the new technologies should be investigated. This would include training and support of K8s, Slinky (slurm-operator), Rancher, cert-manager, and Prometheus. This would help university support staff get up to speed quickly and provide expertise and support to address complex issues that may arise.

- Evaluate and determine any additional equipment cost associated with required hardware for the servers to be used for the service infrastructure, either physical or virtual machines. Based on the pilot installation and results, this would likely include one or two servers for K8s kube-system namespace services, and up to five servers for the Slinky slurm-operator services, including one for login services, one for cert-manager and Prometheus, one for the slurm namespace services, one for the slinky namespace

services, and one for the mariadb namespace services. These server hardware resources (cpu, memory, networking) would need to be determined either by recommendation from SchedMD or by gaining experience in the installation and operation of K8s and Slinky slurm-operator.

- This work also gave rise to the idea and potential for the replacement of the ISAAC NG virtual machine infrastructure with containers and, thereby, potentially saving money by eliminating the need for VMWare or similar hypervisor and hyper-converged infrastructure software and its associated costs.

- Evaluation and creation of a list of criteria, capabilities, and features to test and evaluate to determine a K8s and Slinky slurm-operator installation as production-ready.

- Installation and testing of Slinky slurm-operator version 0.4.0 through 1.0.0 in a phased approach described in the separate list below.

Based on my extensive experience in installing, managing, and directing new and upgraded HPC systems, I suggest the following phased approach for integration of K8s and Slinky slurm-operator:

- Phase I: Develop the list of criteria, capabilities, and features and a plan to test and evaluate with K8s, Slinky slurm-operator, and with the combination of both. This list would be continuously reviewed and updated across the entire phased approach to an integrated K8s and Slinky slurm-operator installation, potentially including SEC and other university research computing directors and staff in the development of this list.

- Phase II: Installation of K8s by itself for staff development, including the Lustre Exascaler CSI storage as the integrated storage and Prometheus for metrics and monitoring, along with identifying friendly users to use the environment and give feedback on the early adoption of K8s. For optimal usability, it is recommended that this installation be paired with Rancher. The pilot used only the K8s command line with no web-based system administrator or user interface installed or tested as part of

the pilot. Note: This would provide key additional information to HPSC staff on the infrastructure resources necessary for a production environment, including Prometheus and Rancher.

- Phase III: Installation of Slinky slurm-operator version 0.4.0 or later to the K8s environment mentioned previously, with a small subset of the HPC cluster compute servers and GPU servers as K8s Worker nodes. This would include using the testing and evaluation plan developed in Phase I to determine whether to move into Phase IV. This would likely be a Slinky slurm-operator hybrid configuration to allow reuse of existing infrastructure, such as any existing LDAP and MariaDB services, to address authentication, authorization, and accounting.

- Phase IV: After successful results from Phase III, a full complement of the GPU servers, including up to all of the generally accessible ISAAC NG GPU servers available to all researchers, could be transitioned into the integrated environment and auto-scaled (which should be tested in Phase III) between K8s and Slurm based on workload.

Bibliography

Imran Ahmed, Gwanggil Jeon, and Francesco Piccialli. 2022. From Artificial Intelligence to Explainable Artificial Intelligence in Industry 4.0: A Survey on What, How, and Where. *IEEE Transactions on Industrial Informatics* 18, 8 (2022), pages 5031–5042. 2, 3

Amazon Web Services. 2016. Netflix Case Study. Retrieved July 22, 2025 from https://aws.amazon.com/solutions/case-studies/netflix-case-study/ 2

Arjun. 2025. Inside eBay Tech Stack and Infrastructure. Retrieved July 22, 2025 from https://appscrip.com/blog/ebay-tech-stack-and-infrastructure/ 2

Auburn University. 2025. Auburn University High Performance and Parallel Computing. Retrieved July 24, 2025 from https://hpc.auburn.edu/hpc/2020_cluster.php 4

Benjamin Black. 2013. Benjimin Black - EC2 Origins. Retrieved July 22, 2025 from hhttps://blog.b3k.us/2009/01/25/ec2-origins.html 2

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. (2021). 1

Chameleon. 2025. Chameleon. Retrieved August 16, 2025 from https://chameleoncloud.org/ 8

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A Survey on Evaluation of Large Language Models. *ACM transactions on intelligent systems and technology* 15, 3 (2024), pages 1–45. 1, 2, 3

Antony Chazapis, Evangelos Maliaroudakis, Fotis Nikolaidis, Manolis Marazakis, and Angelos Bilas. 2024. Running Cloud-native Workloads on HPC with High-Performance Kubernetes. *arXiv preprint arXiv:2409.16919* (2024). 12, 14

Antony Chazapis, Fotis Nikolaidis, Manolis Marazakis, and Angelos Bilas. 2023. Running kubernetes workloads on HPC. In *International Conference on High Performance Computing*. Springer, 181–192. 14

CloudLab. 2025. CloudLab. Retrieved August 16, 2025 from https://cloudlab.us/ 8

CNCF. 2025a. Cloud Native Computing Foundation. Retrieved July 26, 2025 from https://www.cncf.io/ 21

CNCF. 2025b. Extending Kubernetes with Operator Pattern. Retrieved August 14, 2025 from https://kubernetes.io/docs/concepts/extend-kubernetes/operator/ 27

CNCF. 2025c. Kubernetes Releases. Retrieved July 26, 2025 from https://kubernetes.io/releases/ 22

Coreweave. 2023. Introducing SUNK: A Slurm on Kubernetes Implementation for HPC and Large Scale AI. Retrieved July 28, 2025 from https://www.coreweave.com/blog/sunk-slurm-on-kubernetes-implementations 11, 12, 13

Coreweave. 2025. SUNK. Retrieved August 5, 2025 from https://docs.coreweave.com/docs/products/sunk 13, 17

Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, et al. 2019. The Design and Operation of CloudLab. In *2019 USENIX annual technical conference (USENIX ATC 19)*. 1–14. 8

Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. 2016. DevOps. *IEEE software* 33, 3 (2016), pages 94–100. 3

Erik Deumens (host). 2025. Southeastern Conference Research Director Monthly Meeting. Unpublished Discussion, 29 July 2025. 4

FORTH ICS. 2024. High-Performance Kubernetes. Retrieved August 4, 2025 from https://github.com/CARV-ICS-FORTH/HPK 14

GitHub K8s CSI. 2025. Kubernetes CSI Drivers. Retrieved September 23, 2025 from https://kubernetes-csi.github.io/docs/drivers.html 38

Vivian Hafener. 2025. Issue 48. Retrieved September 23, 2025 from https://github.com/SlinkyProject/slurm-operator/issues/48 37

Victor Hazlewood. 2025. slurm-operator: Need description of required compute resources and storage. Retrieved September 30, 2025 from https://github.com/SlinkyProject/slurm-operator/issues/52 36

UTK OIT HPSC. 2025. Academic Computational Resources. Retrieved August 27, 2025 from https://oit.utk.edu/hpsc/academic-resources/ 7

Jensen Huang. 2024a. Jensen Huang Keynote at 2024 NVIDIA GPU Technology Conference. Retrieved July 22, 2025 from https://www.youtube.com/watch?v=Y2F8yisiS6E 7:00-12:00. 2

Jensen Huang. 2024b. Jensen Huang Keynote at 2024 NVIDIA GPU Technology Conference. Retrieved July 22, 2025 from https://www.youtube.com/watch?v=Y2F8yisiS6E 1:57:00-2:00:00. 2

Hugging Face. 2020. The Hugging Face Hub. Retrieved July 22, 2025 from https://huggingface.co/ 3

Indiana University. 2020a. Jetstream 2: Accelerating Science and Engineering On-Demand. Retrieved July 24, 2025 from https://www.nsf.gov/awardsearch/showAward?AWD_ID=2005506 8

Indiana University. 2020b. Jetstream2 Kubernetes. Retrieved July 24, 2025 from https://docs.jetstream-cloud.org/general/kubernetes/ 7

Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S Gunawi, Cody Hammock, et al. 2020. Lessons Learned from the Chameleon Testbed. In *2020 USENIX annual technical conference (USENIX ATC 20)*. 219–233. 8

Gregory M Kurtzer, Vanessa Sochat, and Michael W Bauer. 2017. Singularity: Scientific containers for mobility of compute. *PloS one* 12, 5 (2017), e0177459. 15

Lambda. 2025. Managed and Unmanaged Slurm. Retrieved July 28, 2025 from https://lambda.ai/slurm 17

Linux Foundation. 2021. Apptainer. Retrieved July 24, 2025 from https://apptainer.org/ 6, 15

Louisiana State University. 2025. Louisiana State University High Performance Computing. Retrieved July 24, 2025 from http://hpc.lsu.edu/ 5

Bernard Marr. 2019. *Artificial intelligence in practice: how 50 successful companies used AI and machine learning to solve problems.* John Wiley & Sons, Hoboken, NJ. 2, 8

Bernard Marr. 2023. A Short History Of ChatGPT: How We Got To Where We Are Today. Retrieved July 22, 2025 from https://www.forbes.com/sites/bernardmarr/2023/05/19/a-short-history-of-chatgpt-how-we-got-to-where-we-are-today/ 3

Microsoft. 2025. Microsoft Azure Cyclecloud Workspace for Slurm. Retrieved July 28, 2025 from https://learn.microsoft.com/en-us/azure/cyclecloud/overview-ccws?view=cyclecloud-8 17

Microsoft Corporation. 2016. Adobe and Microsoft Partner in the Azure Cloud to Help Businesses Transform Customer Engagement. Retrieved July 22, 2025 from https://news.microsoft.com/source/2016/09/26/adobe-and-microsoft-partner-in-the-azure-cloud-to-help-businesses-transform-customer-engagement 3

Ron Miller. 2016. How AWS Came To Be. Retrieved July 22, 2025 from https://techcrunch.com/2016/07/02/andy-jassys-brief-history-of-the-genesis-of-aws/ 2

Mississippi State University. 2025. Mississippi State University High Performance Computing Collaboratory. Retrieved July 24, 2025 from https://www.hpc.msstate.edu/ 5

Madhumita Murgia. 2023. Transformers: the Google scientists who pioneered an AI revolution. *Financial Times* 23 (jul 2023), retreived online individually from Factiva. 3

NSF. 2024. National Science Foundation Higher Education Research and Development Expenditures. Retrieved August 18, 2025 from https://ncsesdata.nsf.gov/profiles/site?method=rankingbysource&ds=herd 5

NVIDIA. 2024a. NVIDIA NIM for Developers. Retrieved July 24, 2025 from https://developer.nvidia.com/nim 3, 28

NVIDIA. 2024b. NVIDIA NIM Offers Optimized Inference Microservices for Deploying AI Models at Scale. Retrieved July 24, 2025 from https://developer.nvidia.com/blog/nvidia-nim-offers-optimized-inference-microservices-for-deploying-ai-models-at-scale/ 3

OpenAI. 2025. GPT-5. Retrieved August 7, 2025 from https://openai.com/index/introducing-gpt-5/ 3

POTUS. 2016a. The National Artificial Intelligence Research and Development Strategic Plan. Retrieved July 24, 2025 from https://www.nitrd.gov/pubs/national_ai_rd_strategic_plan.pdf 1

POTUS. 2016b. Preparing for the Future of Artificial Intelligence. Retrieved July 24, 2025 from https://obamawhitehouse.archives.gov/sites/default/files/whitehouse_files/microsites/ostp/NSTC/preparing_for_the_future_of_ai.pdf 1

POTUS. 2019a. Executive Order 13859: Maintaining American Leadership in Artificial Intelligence. Retrieved July 24, 2025 from https://www.federalregister.gov/documents/2019/02/14/2019-02544/maintaining-american-leadership-in-artificial-intelligence 1

POTUS. 2019b. The National Artificial Intelligence Research and Development Strategic Plan: 2019 Update. Retrieved July 24, 2025 from https://trumpwhitehouse.archives.gov/wp-content/uploads/2019/06/National-AI-Research-and-Development-Strategic-Plan-2019-Update-June-2019.pdf 1

POTUS. 2025. America's AI Action Plan. Retrieved July 2025 from https://www.whitehouse.gov/wp-content/uploads/2025/07/Americas-AI-Action-Plan.pdf 1

Nigel Poulton and Pushkar Joglekar. 2025a. *The Kubernetes Book, 2025 Edition*. Nigel Poulton, Ltd, Monee, IL, page 10. 2, 21

Nigel Poulton and Pushkar Joglekar. 2025b. *The Kubernetes Book, 2025 Edition.* Nigel Poulton, Ltd, Monee, IL, page 139. 2

Purdue University. 2020a. Anvil - A National Composable Advanced Computational Resource for the Future of Science and Engineering. Retrieved July 24, 2025 from https://www.nsf.gov/awardsearch/showAward?AWD_ID=2005632&HistoricalAwards=false 8

Purdue University. 2020b. Anvil Composable Subsystem. Retrieved July 24, 2025 from https://www.rcac.purdue.edu/index.php/knowledge/anvil/composable?all=true 7

Purdue University. 2020c. CC* Compute: Private Campus Cloud for Data Analytics and Machine Learning. Retrieved July 24, 2025 from https://www.nsf.gov/awardsearch/showAward?AWD_ID=2005632&HistoricalAwards=false 8

Purdue University. 2020d. Geddes Composable Subsystem. Retrieved July 24, 2025 from https://www.rcac.purdue.edu/index.php/knowledge/geddes 7

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-training. Retrieved July 22, 2025 from https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf 3

Ajeet Raina, Omar Sanseviero, and Nate Raw. 2025. Effortlessly Build Machine Learning Apps with Hugging Face's Docker Spaces. Retrieved July 22, 2025 from https://www.docker.com/blog/build-machine-learning-apps-with-hugging-faces-docker-spaces/ 3, 28

Harish Raj. 2024. How Kubernetes Powers OpenAI's Infrastructure: A 2018–2023 Evolution. Retrieved July 22, 2025 from https://www.linkedin.com/pulse/how-kubernetes-powers-openais-infrastructure-20182023-harish-raj-m-kkocc/ 3

Rancher by SUSE. 2025. Rancher: Enterprise Kubernetes Management. Retrieved September 16, 2025 from https://www.rancher.com/ 43

Rancher Labs. 2020. Rancher: Enterprise Kubernetes Management Platform and Software. Retrieved July 26, 2025 from https://www.rancher.com/ 27

SchedMD. 2020a. Slurm Training Documentation. pages 19-20, Last accessed July 28, 2025. 17, 18

SchedMD. 2020b. Slurm Training Documentation. pages 18-19, Last accessed July 28, 2025. 18

SchedMD. 2025a. Slinky. Retrieved August 4, 2025 from https://slinky.ai/ 11, 13

SchedMD. 2025b. Slinky Project. Retrieved August 4, 2025 from https://github.com/slinkyproject 11, 13, 35

SchedMD. 2025c. Slinky slurm-operator version 0.3.1 Quickstart Guide. Retrieved September 22, 2025 from https://github.com/SlinkyProject/slurm-operator/blob/v0.3.1/docs/quickstart.md 35

SchedMD. 2025d. Slurm 25.05 release notes. Retrieved August 5, 2025 from https://slurm.schedmd.com/release_notes.html 12

SchedMD. 2025e. Slurm History. Retrieved July 28, 2025 from https://www.schedmd.com/about-schedmd/slurm-history/ 16

SchedMD. 2025f. slurm-k8s-bridge. Retrieved August 5, 2025 from https://github.com/SlinkyProject/slurm-bridge 12, 29

Texas A&M University. 2025. Texas A&M University High Performance Research Computing. Retrieved July 24, 2025 from https://hprc.tamu.edu/kb/ 5

University of Alabama. 2025. The University of Alabama OIT Research Computing. Retrieved July 24, 2025 from https://hpc.ua.edu/ 4

University of Arkansas. 2025. Arkansas High Performance Computing Center. Retrieved July 24, 2025 from https://hpc.uark.edu/ 4

University of Florida. 2025a. HiPerGator. Retrieved August 17, 2025 from https://www.rc.ufl.edu/hipergator/ 4

University of Florida. 2025b. University of Florida Research Computing. Retrieved July 24, 2025 from https://www.rc.ufl.edu/ 4

University of Georgia. 2025. Georgia Advanced Computing Resource Center. Retrieved July 24, 2025 from https://gacrc.uga.edu/ 5

University of Kentucky. 2025. University of Kentucky Center for Computational Sciences. Retrieved July 24, 2025 from https://www.ccs.uky.edu/ 5

University of Memphis. 2023. The University of Memphis High Performance Computing. Retrieved July 24, 2025 from https://www.memphis.edu/hpc/current-faculty-students.php 4

University of Mississippi. 2025. Mississippi Center for Supercomputing Research. Retrieved July 24, 2025 from https://mcsr.olemiss.edu/supercomputers/magnolia/ 5

University of Missouri. 2025. University of Missouri Information Technology Research Support Solutions The Mill Cluster. Retrieved July 24, 2025 from https://itrss.mst.edu/cluster/mill/ 5

University of Oklahoma. 2025. OU Supercomputing Center for Education & Research. Retrieved July 24, 2025 from https://www.ou.edu/oscer 5

University of South Carolina. 2025. University of South Carolina High Performance Computing Resources. Retrieved July 24, 2025 from https://sc.edu/about/offices_and_divisions/division_of_information_technology/rc/resources/index.php 5

University of Tennessee, Chattanooga. 2025. University of Tennessee, Chattanooga Research Institute Resources. Retrieved July 31, 2025 from https://www.utc.edu/research/research-institute/facilities-equipment-and-other-resources 4

University of Texas. 2025. Texas Advanced Computing Center Lonestar6. Retrieved July 24, 2025 from https://tacc.utexas.edu/systems/lonestar6/ 5

USC. 2019. Running a Job on HPC using Slurm. Retrieved July 28, 2025 from https://web.archive.org/web/20190306044130/https://hpcc.usc.edu/support/documentation/slurm/ 16

UTK OIT. 2022. Running Jobs on ISAAC-NG. Retrieved July 24, 2025 from https://oit.utk.edu/hpsc/isaac-open-enclave-new-kpb/running-jobs-new-cluster-kpb/ 4, 5

UTK OIT HPSC. 2024. AI Across Tennessee Symposium. Retrieved July 22, 2025 from https://oit.utk.edu/hpsc/ai-across-tennessee-symposium-2024/ 4

Vanderbilt University. 2025. ACCRE Technical Details. Retrieved July 24, 2025 from https://www.vanderbilt.edu/accre/technical-details/ 5

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. *Advances in Neural Information Processing Systems* 30 (2017). 2, 3

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. (2022). 1

Tim Wickberg. 2024. Slurm and/or/vs Kubernetes. Retrieved July 28, 2025 from https://slurm.schedmd.com/SC23/Slurm-and-or-vs-Kubernetes.pdf 11, 12, 17, 21, 28, 29

Tim Wickberg. 2025. Slinky: Slurm in Kubernetes, Performant AI and HPC Workload Management. Retrieved July 28, 2025 from https://slurm.schedmd.com/MISC25/Slinky-KubeConEurope2025.pdf 6, 11, 12, 21, 28

Wikipedia. 2025. Slurm Workload Manager. Retrieved July 28, 2025 from https://en.wikipedia.org/wiki/Slurm_Workload_Manager 16

Banghua Wu, Menglong Hu, Shuaibing Qin, and Jinliang Jiang. 2024. Research on fusion scheduling based on Slurm and Kubernetes. In *International Conference on Algorithms,*

High Performance Computing, and Artificial Intelligence (AHPCAI 2024), Vol. 13403. SPIE, 476–485. 2, 6, 12, 13

George Zervas, Antony Chazapis, Yannis Sfakianakis, Christos Kozanitis, and Angelos Bilas. 2022. Virtual clusters: isolated, containerized HPC environments in kubernetes. In *International Conference on High Performance Computing*. Springer, 347–357. 14

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. (2023). 1

Appendices

A Copyrights and Licenses

A.1 Slurm

A.2 Slinky

License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

A.3 Kubernetes

Kubernetes software and documentation images are Copyright © 2025 Cloud Native Computing Foundation which is part of the Linux Foundation.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

A.4 Kubernetes documentation

Kubernetes documentation is licensed under the Creative Commons Attribution 4.0 International license. A copy of the license can be obtained at:

https://github.com/kubernetes/website/blob/main/LICENSE

This license includes fair use of Kubernetes documentation graphics used in derivative figures. Figures using Kubernetes documentation graphics are copyright Victor Hazlewood, except those figures as reproduced by permission or by license and indicated by a footnote in the figure caption.

B AI and Machine Learning Courses at UTK Fall 2025

University of Tennessee, Knoxville - Fall 2025 Courses with AI, Artificial Intelligence, or Machine Learning in the course title				
Course Title	**Dept**	**Course**	**Registered**	**Description**
Introduction to the World of AI	Artificial Intelligence	101	50	Introduction to foundational concepts, techniques, and applications of Artificial Intelligence (AI) relevant for all disciplines – especially across non-computer science fields. Explores the history and current scope of AI, data sources and tools, and fundamental components of AI solutions. Special attention will be placed on the strengths and weaknesses of the methods as well as on identifying bias, social impacts, and other ethical considerations of AI. Students will gain experience through hands-on activities using no-code AI platforms. Satisfies Volunteer Core Requirement: (EI) (QR)
Introduction to the World of AI	Artificial Intelligence	101	70	"
Introduction to the World of AI	Artificial Intelligence	101	75	"
Introduction to the World of AI	Artificial Intelligence	101	50	"
Introduction to the World of AI	Artificial Intelligence	101	50	"
AI, Ethics, and Legal Frameworks	Artificial Intelligence	201	35	This course probes the vital junctures of artificial intelligence, ethics, and legal considerations. Throughout their studies, students will explore the ethical challenges posed by emerging AI technologies, understand the societal ramifications of AI decisions, and familiarize themselves with the global legal frameworks that oversee AI applications. As part of the curriculum, students will delve into ethical principles central to AI design and deployment. They will also address concerns related to bias, fairness, and transparency in AI systems. Additionally, discussions will touch upon the crucial aspects of privacy and data rights. An in-depth look into regulatory landscapes will offer insights into AI governance. By the end of this course, students will be primed to approach AI solutions holistically, bearing in mind both the ethical and legal dimensions. (RE) Prerequisite(s): AI 101 or AI 102
Natural Language Processing and Conversational AI	Artificial Intelligence	301	21	This course provides a comprehensive study of Natural Language Processing (NLP) and how it empowers the creation of conversational AI agents. Students will begin with an understanding of the fundamentals of NLP and linguistics. Building upon that, the course will guide them through the process of constructing chatbots and conversational interfaces. They will also engage in sentiment analysis and delve into the intricacies of text classification. Advanced topics, such as the mechanics of dialogue systems and the nuances of machine translation, will further enrich their learning journey. By the course's conclusion, students will not only have hands-on experience in crafting conversational AI applications but will also possess a profound grasp of the foundational NLP techniques that drive them. (RE) Prerequisite(s): AI 101 or AI 102
AI-based Data Handling and Visualization	Artificial Intelligence	302	18	Data is the lifeblood of AI. In this course, students will learn how to efficiently handle, process, and visualize data using AI-driven techniques. The key areas of study encompass the use of AI tools for data preprocessing and cleaning, advanced data analysis using AI algorithms, visualization techniques for high-dimensional data, and exploring real-world applications and case studies. Throughout the course, students will gain hands-on experience with popular AI data tools and libraries, preparing them for complex AI-driven data tasks. (RE) Prerequisite(s): AI 101 or AI 102
AI for Cybersecurity	Artificial Intelligence	311	13	As the landscape of cyber threats continues to change, the demand for innovative AI-driven solutions becomes paramount. In this course, students will journey through the intricate relationship between artificial intelligence and cybersecurity. They'll begin with an introduction to the foundational principles of cybersecurity and the challenges that professionals face in this domain. As they progress, they'll familiarize themselves with the AI tools and algorithms tailored for threat detection and mitigation. Furthermore, through case studies, students will gain insights into AI-driven cyberattacks and the corresponding defense strategies employed to counteract them. The curriculum is further enriched with hands-on projects that emulate real-world cybersecurity scenarios. Upon completing the course, students will be adept at designing and executing AI strategies to enhance cybersecurity infrastructures. (RE) Prerequisite(s): AI 101 or AI 102
AI and Data-Driven Clinical Solutions and Projects	Artificial Intelligence	351	4	Students will embark on a comprehensive exploration of the transformative potential of artificial intelligence and data science in the medical and healthcare sectors. The course delves into the practical application of AI algorithms and data science tools for diagnosing diseases, predicting patient outcomes, optimizing treatment plans, and enhancing patient care. Students will be introduced to real-world clinical problems and collectively explore potential AI and data-driven solutions for those problems. This collaborative approach ensures that all students gain exposure to various clinical scenarios. It also lays the groundwork for individual project work, which becomes the focal point in subsequent advanced courses like AI 451. (RE) Prerequisite(s): AI 101 or DATA 101
Applied AI for Research and Applications	Artificial Intelligence	401	18	Detailed study of concepts, techniques, and applications of Artificial Intelligence (AI) relevant for all disciplines – especially across non-computer science fields. Explores the history and current scope of AI, data sources and procedures for attaining and working with data, and fundamental components of AI solutions. Special attention will be placed on the strengths and weaknesses of the methods as well as on identifying bias, social impacts, and other ethical considerations of AI. Introduces students to AI-relevant programming through hands-on coding projects. This is an undergraduate and graduate course (AI 501) taught concurrently, where graduate students will have additional requirements and assignments. Credit Restriction: students cannot receive credit for both AI 401 and AI 501.

Applied AI for Research and Applications	Artificial Intelligence	501	5	no course description
Sociology, Science Fiction Film, and Artificial Intelligence	Cinema Studies	419	11	Examines how science-fiction films center on the tension between society and technology; focuses on artificial intelligence to understand the persistence of social problems and issues, via important contributions to sociology and social theory. (See SOCI 419) Grading Restriction: Letter grade only
Introduction to Machine Learning	Computer Science	325	69	Machine learning is concerned with computer programs that automatically improve their performance through experience. This course covers the theory and practice of machine learning from a variety of perspectives. We cover topics such as clustering, decision trees, neural network learning, statistical learning methods, Bayesian learning methods, dimension reduction, kernel methods, and reinforcement learning. Programming assignments include implementation and hands-on experiments with various learning algorithms. (RE) Prerequisite(s): ECE 313 or ECE 317 or MATH 323 with a grade of C or better; and MATH 251 or MATH 257 with a grade of C or better.
Machine Learning	Computer Science	522	59	Theoretical and practical aspects of machine learning techniques related to pattern recognition. Statistical methods studied include Bayesian and linear classifiers, support vector machines, neural networks, and unsupervised learning. Syntactic methods include grammatical inference, string matching and Markov chains. Ensemble methods include random forests, adaptive boosting, and classifier fusion
Introduction to Artificial Intelligence	Computer Science	423	61	Foundational ideas and developments in artificial intelligence. Problem solving and search, knowledge representation and reasoning, decision-making under uncertainty, machine learning, and multi-agent systems. (RE) Prerequisite(s): COSC 302 or COSC 307 with a grade of C or better; MATH 251 or MATH 257 with a grade of C or better; and ECE 313 or ECE317 or MATH 323 or STAT 251 with a grade of C or better. Comment(s): Prior knowledge may satisfy prerequisites with consent of instructor.
Artificial Intelligence	Computer Science	523	32	"
Artificial Intelligence	Computer Science	523	100	Theoretical and applied aspects of artificial intelligence. Course topics include problem solving and search, knowledge representation and reasoning, decision-making under uncertainty, machine learning, and multi-agent systems. Recommended Background: Programming, data structures and algorithms, linear algebra, probability theory.
AI in Power	Computer Science	593	4	Artificial Intelligence and Biosensors
AI for Human Performance Analysis	Computer Science	650	1	no course description
AI & Human Development	Educational Leadership/ Policy Studies	495	2	no course description
AI & Human Development	Educational Leadership/ Policy Studies	595	8	no course description
Digital Publication in AI Age	English	462	19	MANDATORY FIRST WEEK ONLINE ACCESS This course is fully online with weekly activities and submissions required throughout the course of the semester. This class does not have an assigned time, but the work must be completed by the given deadlines. The instructor is accessible through email and online video conferences
Introduction into Machine Learning	Material Sciences	404	2	This course is designed to provide the fundamental background of ML methods as applied to practical problems of materials discovery, characterization, and optimization. It covers the basics of the classification, regression, and dimensionality reduction methods, combined physics- and ML based workflows, and introduces the concepts of active learning, experimental planning, and causal learning as necessary elements of decision making. The emphasis is made on low-code hands-on practice.
Introduction to Machine Learning	Material Sciences	504	22	This course is designed to provide the fundamental background of ML methods as applied to practical problems of materials discovery, characterization, and optimization. It covers the basics of the classification, regression, and dimensionality reduction methods, combined physics- and ML based workflows, and introduces the concepts of active learning, experimental planning, and causal learning as necessary elements of decision making. The emphasis is made on low-code hands-on practice.
Math methods/machine learning	Mathematics	519	11	This course explores the mathematical foundations critical to machine learning, offering a comprehensive examination of both theoretical concepts and practical applications. Key mathematical topics covered include linear equations, regularization, singular value decomposition, and iterative optimization algorithms. The course also examines machine learning methods such as kernel methods, manifold learning, clustering, and decision tree-based approaches. Designed for students with a background in undergraduate data science and programming, this course equips participants with the tools to analyze and implements machine learning models, effectively bridging theory and practice.
Reasoning & Philosophy of AI	Philosophy	400	7	no course description
Reasoning & Philosophy of AI	Philosophy	528	8	no course description
Sociology, Science Fiction Film, and Artificial Intelligence	Sociology	419	3	no course description

Vita

Victor Gene Hazlewood received his Bachelor of Science degree in Computer Science from Texas A&M University in August 1988. Victor then worked in high performance computing in roles from system administrator at General Dynamics (1988) and Texas A&M University Supercomputer Center (1989-1996), Manager (1996-2002) and Information Security Officer (2004-2006) at San Diego Supercomputer Center, Director of Information Assurance at Strategic Data Systems (2002-2003) and obtaining the ISC^2 Certified Information Systems Security Professional credential (2002), senior cybersecurity analyst at Oak Ridge National Laboratory (2006-2008), senior systems analyst, chief information security officer and chief operating officer at the University of Tennessee/Oak Ridge National Laboratory Joint Institute for Computational Sciences (2008-2020), interim advanced computing facility director (2019), and, finally, Associate CIO and Director of High Performance & Scientific Computing (2020-2025) at University of Tennessee, Knoxville Office of Information Technology/Office of Innovative Technology. While at University of Tennessee, Knoxville Victor has worked on his Master of Computer Science degree while working full time until he retired after 17 years in May 2025, culminating with finishing the requirements for Master of Computer Science with this thesis.