



Dflare.AI Spec Sheet

2024

Table of Contents

1.	Introduction	2
2.	DFlare.AI Essentials Includes	2
3.	DFlare.AI Cluster Manager Includes	5
4.	DFlare.AI Plus Includes.....	6

1. Introduction

Coredege DFlare.AI Platform services help Governments and Countries achieve AI Sovereignty By offering them increased control over data location and computing infrastructure and how they manage it, enabling strict governance of sensitive information, ensuring compliance with local regulations, and providing the flexibility to customize AI solutions according to specific national needs.

DFlare.AI Platform comes in 3 major editions. Coredege offers these editions to partners and enterprises as per the requirements with customization to offer GPU as a service or AI as a service.

1. DFlare.AI Essentials
2. DFlare.AI Cluster Manager
3. DFlare.AI Plus

2. DFlare.AI Essentials Includes

1. Central IAM & Multi-Tenancy

- **Comprehensive Control:** Adheres to industry standards, ensuring that only authorized personnel access certain system components.
- **Role-Based Access Control (RBAC):** Manages permissions efficiently, minimizing security risks.
- **Data Governance:** Maintains strict compliance with regulatory standards, enhancing data security and integrity.

2. Bare Metal GPU Manager

- **Audit and Compliance:** Provides comprehensive logging and monitoring capabilities to ensure transparency in actions and decisions made within the system, meeting regulatory and organizational compliance standards.
- **Role-Based Access Control (RBAC):** Allows granular control over who can access and manage BareMetal resources, enabling secure multi-tenant environments and safeguarding critical infrastructure.
- **Provisioning Automation:** Automates the OS setup and driver configuration of BareMetal servers, streamlining the deployment process for faster resource availability and reducing manual intervention.
- **Health Monitoring:** Continuously tracks the performance and health of BareMetal hardware, providing real-time alerts and diagnostics to ensure system reliability and uptime.

3. Storage Manager (PNFS / PFS)

- **Tenant Isolation:** The platform supports multi-tenant environments by ensuring complete isolation between tenants. Each tenant has its own dedicated volumes, and resources, preventing data overlap or leakage between different tenants.
- **Role-Based Access Control (RBAC):** Allows fine-grained access control to storage resources, ensuring that only authorized users can create, modify, or delete volumes. RBAC policies are set per tenant, ensuring different teams or departments within an organization can manage their storage independently.
- **Quota Management:** Each tenant can have specific storage quotas, limiting the total amount of storage they can consume. Administrators can define and enforce storage limits on a per-tenant basis, ensuring that tenants do not exceed their allocated resources.
- **Tenant Usage Monitoring:** Provides real-time monitoring and usage analytics for each tenant, enabling both tenants and administrators to track storage usage, performance metrics, and costs, ensuring efficient resource allocation.
- **Dynamic Provisioning with Kubernetes:** The CSI driver automates the creation and deletion of Persistent Volumes (PVs) and Persistent Volume Claims (PVCs) within Kubernetes environments. Users can create PVCs through Kubernetes YAML manifests, and the platform provisions the required storage volume in target storage backend automatically.
- **Volume Snapshots and Restores via CSI:** The CSI driver supports Kubernetes VolumeSnapshot functionality, enabling users to take snapshots of PVCs and restore them when needed. This is useful for backup, data protection, and disaster recovery scenarios.
- **Multi-Attach and Read-Write Access:** Supports ReadWriteMany (RWX) access modes, enabling multiple pods to mount and access the same storage volume simultaneously. This is critical for stateful applications running in containerized environments that require shared access to storage.
- **Storage Attached in Server:** Storage and disk profile management for disk attached in servers.

4. Network Manager for Ethernet

- **VPC:** Enables the creation and management of VLANs for network segmentation, enhancing security and performance in multi-tenant environments.
- **Dynamic IP Management:** Handles automatic IP allocation and configuration for devices, simplifying network management and improving scalability.

5. Network Manager for InfiniBand

- **Advanced Fabric Management:** Manages the high-speed fabric network, providing ability to create partition keys, fault management, and optimizing InfiniBand network utilization.

6. Centralized Observability

- **Robust Metrics:** Provides continuous oversight with its monitoring capabilities, aiding in timely issue resolution and relevant metrics where you need them.
- **Centralized Logging:** Aggregates logs from all sources aiding in timely issue resolution.

7. Object Storage Manager

- **Bucket LCM:** Provide capability of managing buckets on Object Storage with Create / Update / Delete actions on bucket.
- **Access Token management:** allowing configurable access for advanced users
- Define simplified UX for consuming buckets and Object storage locations

8. Metering and Show back Service

- **FinOps:** Tools for cost management to enhance budget adherence and efficiency.
- **Usage Tracking:** track all the resources consumed by the user.
- **Show back:** Show all the current estimated charges for resource consumption to user on portal. This enabled detailed view for users to track and plan future infrastructure spending accordingly.

3. DFlare.AI Cluster Manager Includes

1. Managed Kubernetes

- Kubernetes LCM: Provide capability of creating and managing k8s cluster
- Node Management: Streamlines configuration and management of compute nodes to suit different roles and tasks.
- Kubernetes Addon: Provide capability of creating and managing CNI and CSI for Kubernetes clusters.

2. Managed Slurm

- **Slurm Cluster LCM:** Manage lifecycle of Slurm Cluster with creation of cluster and management.
- Node Grouping and Partitioning: Enables the creation of node partitions for different job types or user groups, improving resource management and workload segregation.
- Real-time Job Monitoring: Provides a centralized interface to monitor running jobs, resource utilization, and job performance in real time, ensuring better visibility for administrators.

4. DFlare.AI Plus Includes

1. Training-as-a-service

- **Experiment Tracking:** Tracking and logging of experiments.
- **Model Deployment Integration:** If model deployment is part of the workflow, ensure integration with deployment platforms. This might involve coordinating the deployment process with the completion of successful training jobs.
- **Security Measures:** Security measures to protect data, manage user access, and secure communication within the Slurm cluster.
- **Scaling and Auto-scaling:** Scaling the Slurm cluster based on demand. Auto-scaling mechanisms can be implemented to dynamically adjust the number of compute nodes based on workload.
- **Monitoring and Logging:** Monitoring tools to track the performance of Slurm nodes, job progress, and resource utilization.

2. Inferencing-as-a-service

- **Multi-Model and Multi-Framework Support:** Triton supports multiple model frameworks (like TensorFlow, PyTorch, ONNX, etc.) and allows serving different models concurrently, making it highly versatile for various use cases.
- **Efficient Inference Execution:** Triton optimizes inference through features like dynamic batching and model versioning, ensuring low-latency predictions and efficient resource utilization.
- **Model Management:** Easily manage model lifecycles (load/unload models dynamically) and handle multiple versions of the same model for A/B testing and production deployment flexibility.
- **Dynamic Scaling Based on Load:** The platform automatically scales inference server instances up or down based on real-time traffic and load, ensuring resource efficiency and cost optimization.
- **Elastic Resource Management:** Seamless adjustment to spikes or drops in inference requests by scaling infrastructure in cloud-native environments like Kubernetes, preventing bottlenecks and ensuring smooth performance under varying loads.
- **Horizontal Pod Autoscaling (HPA):** Integration with Kubernetes allows auto-scaling based on CPU, memory, and custom metrics (like inference request rate), ensuring the system adapts dynamically to workload demands.
- **Standardized Model Inference API:** KServe provides standardized REST/gRPC API endpoints for model serving, allowing seamless interaction and invocation of models for inferencing across diverse client applications.
- **Model Monitoring and Logging:** KServe APIs allow built-in logging and monitoring of requests, responses, and performance metrics, offering detailed insights into model behavior and helping with debugging or optimization.

- **Multi-Model Management and Versioning:** KServe based management with multiple models, their versions, and can dynamically switch between models, making it easier to manage production-grade AI systems without manual intervention.

3. Container Registry Service

- **Centralized Storage:** The Container Registry offers a centralized repository for container images, allowing organizations to store all their AI models and associated applications in a private registry. This centralized approach facilitates better management, version control, and accessibility.
- **Version Control and Management:** With robust version control capabilities, the registry service allows users to manage different versions of their container images effectively. This feature is crucial for maintaining consistency across deployments and rolling back to previous versions if needed.
- **Secure Access and Authentication:** The service includes advanced authentication and authorization mechanisms to control access to container images. Role-based access control (RBAC) ensures that only authorized users and systems can access or modify the stored containers.

4. Serverless GPU

- **Dynamic GPU Provisioning:** Automatically allocates and deallocates GPU resources based on the needs of the job or application. This means GPUs are available when required and released when idle, optimizing costs and resources.
- **Serverless Abstraction:** Developers and data scientists can submit multi-GPU or single-GPU tasks without worrying about the infrastructure, provisioning, or GPU management. The serverless framework automatically handles it in the background.
- **Granular Resource Allocation:** Allows using part of a GPU (fractional GPU) or specific GPU types (e.g., H100, GB200) based on the workload requirements, optimizing resource usage.
- **Automatic Parallelization for Multi-GPU:** Supports automatic distribution of large model training or compute-intensive tasks across multiple GPUs seamlessly, ensuring faster training times and scalability.
- **Scaling Down for Single-GPU Jobs:** Efficiently handles single-GPU tasks by scaling down and running applications on one GPU when required. This is ideal for inference jobs or smaller training tasks that don't need multiple GPUs.
- **Seamless Job Scaling:** The system can scale up or down based on the workload, enabling developers to run multiple jobs concurrently on different GPUs without manual resource management.
- **Optimized Resource Utilization:** The serverless model ensures optimal resource utilization by distributing workloads across GPUs based on demand, reducing idle time and enhancing cost savings, especially for sporadic or unpredictable workloads.
- **Pay-per-Use Pricing Model:** Charges users only for the exact GPU usage and execution time, making it highly cost-effective compared to dedicated GPU clusters that require constant provisioning.

5. ML Workspaces

- **On-Demand Compute:** The serverless architecture automatically provisions the required compute resources (CPU/GPU/TPU) as users run experiments, scaling up or down based on demand, ensuring cost-efficiency and optimized resource usage.
- **Pay-per-Use Model:** Users only pay for the resources they consume, with automatic resource release after inactivity or job completion, reducing the overall cost of running ML experiments.
- **Pre-configured Environments:** The workspace comes with pre-configured deep learning libraries like TensorFlow, PyTorch, MXNet, and Keras, allowing users to quickly start building models without manual library installation or configuration.
- **Custom Library Integration:** Users can integrate and install custom or proprietary deep learning libraries as part of their workspace setup, enabling specific use cases, model types, or optimizations for specific hardware architectures (e.g., CUDA, cuDNN).
- **Version Control for Libraries:** Provides version control for installed libraries, enabling users to switch between different versions of deep learning frameworks and ensuring compatibility with existing projects and models.
- **Interactive Notebook Environment:** Provides an interactive Jupyter-based notebook environment, allowing users to write and execute Python code, visualize data, and run experiments within a single interface.
- **Notebook Autoscaling:** Automatically allocates resources based on notebook demand, scaling compute resources in real-time as models are trained, data is processed, or when multiple users access the same workspace.
- **Pre-built Workspace Templates:** Users can start with pre-built templates for specific tasks (e.g., NLP, computer vision, reinforcement learning) with the necessary tools and libraries already installed, reducing the setup time.
- **User-defined Environments:** Each user or team can define their environment with specific versions of libraries, frameworks, and tools tailored to their workflows, ensuring that their workspace meets the exact needs of their ML experiments.
- **Containerized Workspaces:** Uses containerization (e.g., Docker, Kubernetes) to isolate user environments, ensuring reproducibility and preventing dependency conflicts across different projects or teams.

6. Model Binary Repository (GIT LFS)

- **Git LFS for Large Files:** Git LFS replaces large files in your Git repository with lightweight references, while storing the actual content on a remote storage backend. This allows teams to work efficiently without cluttering the Git repository with heavy binaries such as machine learning models and datasets.
- **Support for ML Models and Datasets:** Git LFS supports storing machine learning models, weights, datasets, and other binary files that are too large for standard Git repositories, making it ideal for AI/ML projects.
- **Scalable Storage Backend:** The actual binary files are stored in a scalable object storage backend, ensuring that repositories can handle large-scale model files without performance degradation.

- **Access Control and Permissions:** Integration with Git allows you to enforce access control policies, ensuring that only authorized users can upload, modify, or download models from the repository. This is critical in enterprise environments where security and data governance are important.

7. Cluster Protection Service

- **Backup of Kubernetes Cluster State:** Backup to ensure that the entire Kubernetes cluster state, including configuration, resources, and control plane components (such as etcd, RBAC, and API resources), is backed up. This guarantees that the control plane can be fully restored in case of a disaster.
- **Scheduled and On-Demand Backups:** The service supports both scheduled backups (daily, weekly, or based on custom intervals) and on-demand backups, giving administrators flexibility in how frequently they want to back up the cluster control plane.
- **Versioned Control Plane Snapshots:** Each backup represents a snapshot of the control plane at a specific point in time. These snapshots are versioned and stored securely, enabling rollback to earlier versions in the event of misconfigurations, upgrades, or failures.
- **Backup of Application Resources:** the system backs up all Kubernetes resources, including deployments, services, ConfigMaps, Secrets, and other custom resources that define applications. This ensures that all manifests required to redeploy the application are securely stored.
- **Cross-cluster Migration:** the system can be used to migrate workloads and applications between clusters by restoring backups into a different Kubernetes environment. This is particularly useful for DR testing, upgrades, or moving between development, staging, and production environments.
- **Rollback to Earlier States:** In case of failed deployments, updates, or cluster misconfigurations, the system allows rollbacks to previous states of the cluster, providing an added layer of resilience and minimizing downtime.
- **Centralized Backup Management:** A user-friendly control panel provides centralized management for all backup and restore operations. Administrators can view the status of backups, schedule future backups, and initiate restores through the control panel.

8. Cluster Certification

- **Load and Stress Testing of API Endpoints:** The service benchmarks the robustness of Kubernetes APIs by subjecting the API server to high levels of load and stress. This ensures that the API server can handle increased traffic, sudden spikes, and complex queries without crashing or experiencing significant performance degradation.
- **Error Handling and Recovery Testing:** Tests are performed to check how well the API server handles incorrect inputs, invalid requests, and other error scenarios. This helps to ensure that the API server can recover gracefully from failures and does not expose vulnerabilities or cause crashes due to malformed requests.
- **Node and Pod Scaling Tests:** The service benchmarks the cluster's ability to scale up and down based on workload demand. It includes node scaling tests

(adding/removing nodes dynamically) and pod scaling tests (horizontal pod autoscaling) to ensure that the cluster can handle large workloads while maintaining performance.

- **Resource Utilization Monitoring:** CPU, memory, and I/O utilization are measured during benchmarking tests to ensure that the cluster is using resources efficiently. Performance degradation due to resource constraints or bottlenecks is identified and optimized.
- **Latency and Throughput Measurement:** The service measures the response times of critical operations like creating pods, scheduling workloads, and communicating between services. Latency and throughput metrics help assess how quickly the cluster can respond to API requests under varying load conditions.
- **CIS Kubernetes Benchmark Validation:** The service validates the Kubernetes cluster against the CIS Kubernetes Benchmark, a widely used security standard. It checks for over 120 recommended security configurations, such as securing the API server, controller manager, worker nodes, and network policies. Some key checks include:
 - **API Server Security:** Ensuring that the API server is configured securely (e.g., disabling anonymous access, enforcing RBAC, using TLS for communication).
 - **Worker Node Hardening:** Verifying that the kubelet and other worker node services are hardened and adhere to security best practices.
 - **Network Policies and Pod Security:** Ensuring that network policies are in place and that pods are restricted from communicating with each other unnecessarily.

9. Load Balancer as a Service

- **Layer 4 and Layer 7 Load Balancing:** Supports both Layer 4 (TCP/UDP) and Layer 7 (HTTP/HTTPS) load balancing to ensure optimal routing for low-level traffic as well as application-layer traffic, accommodating diverse workloads from web applications to data-intensive AI models.
- **Workload-Specific Routing:** Supports advanced routing logic based on the type of workload (e.g., AI inference vs. training jobs). It can prioritize certain tasks (like inference) to low-latency, high-throughput nodes with GPUs, while distributing training jobs to nodes that can afford higher computational loads.
- **Health Checks and Monitoring:** Continuously monitors the health of both containerized and bare-metal services through periodic health checks. These checks ensure that only healthy nodes receive traffic, while unresponsive nodes are temporarily removed from the load-balancing pool.
- **Dynamic Scaling:** The load balancer works with container orchestration platforms (e.g., Kubernetes Horizontal Pod Autoscaler) to automatically scale up or down based on traffic load. When demand increases, new containers or services are instantiated, and the load balancer adjusts traffic distribution accordingly.
- **Bare Metal Resource Scaling:** In bare-metal environments, the load balancer can integrate with resource management tools to dynamically allocate GPU resources or add/remove physical servers based on computational demand.

10. Model catalogue and Marketplace

- **Wide Range of Models:** The catalogue includes a variety of pre-trained models across different domains such as image classification, object detection, natural language processing, speech recognition, recommendation systems, and generative models.
- **Model Formats:** Supports multiple model formats such as ONNX, TensorFlow, PyTorch, Hugging Face Transformers, and more, ensuring compatibility with different AI frameworks and platforms.
- **Model Architectures:** Offers popular model architectures like BERT, GPT, ResNet, YOLO, VGG, MobileNet, T5, etc., allowing users to choose models that best fit their application.
- **Ready-to-Deploy:** Models can be downloaded and deployed directly into production environments, supporting a wide range of deployment targets (cloud, edge, on-premise).
- **Application Categories:** AI applications are categorized based on functionality such as image recognition, natural language understanding, recommendation engines, predictive analytics, voice assistants, and generative AI. This allows users to easily find relevant applications.
- **Pre-built AI Modules:** Modules such as chatbots, virtual assistants, facial recognition, sentiment analysis, or recommendation systems can be easily integrated into existing systems.
- **Ready-to-Use Applications:** The marketplace includes turnkey AI solutions that are ready to be deployed with minimal setup. Examples include AI-powered customer support tools, inventory management solutions, and AI-based fraud detection systems.
- **No-Code/Low-Code Solutions:** Many applications are designed for non-technical users, providing drag-and-drop interfaces or no-code workflows to set up and customize AI models without writing complex code.



About CoreEdge

CoreEdge is a leading provider of cloud-native and AI-driven solutions, leveraging a cutting-edge approach to deliver high-performing cloud infrastructure, Kubernetes, and hyper-converged technologies. With its CoreEdge Cloud Suite (CCS), the company enables businesses to build and scale multi-cloud environments while simplifying management through automation and integration. CoreEdge supports organizations in developing cloud-native applications, modernizing legacy systems, and optimizing infrastructure performance. As a trusted partner to major telco, SMEs, & Government, CoreEdge drives digital transformation, helping businesses innovate and thrive in a rapidly evolving technological landscape.

Noida

Office No. 2, 6th Floor,
Tower-B, Embassy Galaxy Business Park,
A-44 & 45, Sector 62, Noida,
Uttar Pradesh 201309

Bangalore

BHIVE Workspace,
No. 467/468 Shri Krishna Temple
Road, Indira Nagar 1st stage,
Bengaluru, Karnataka - 560038