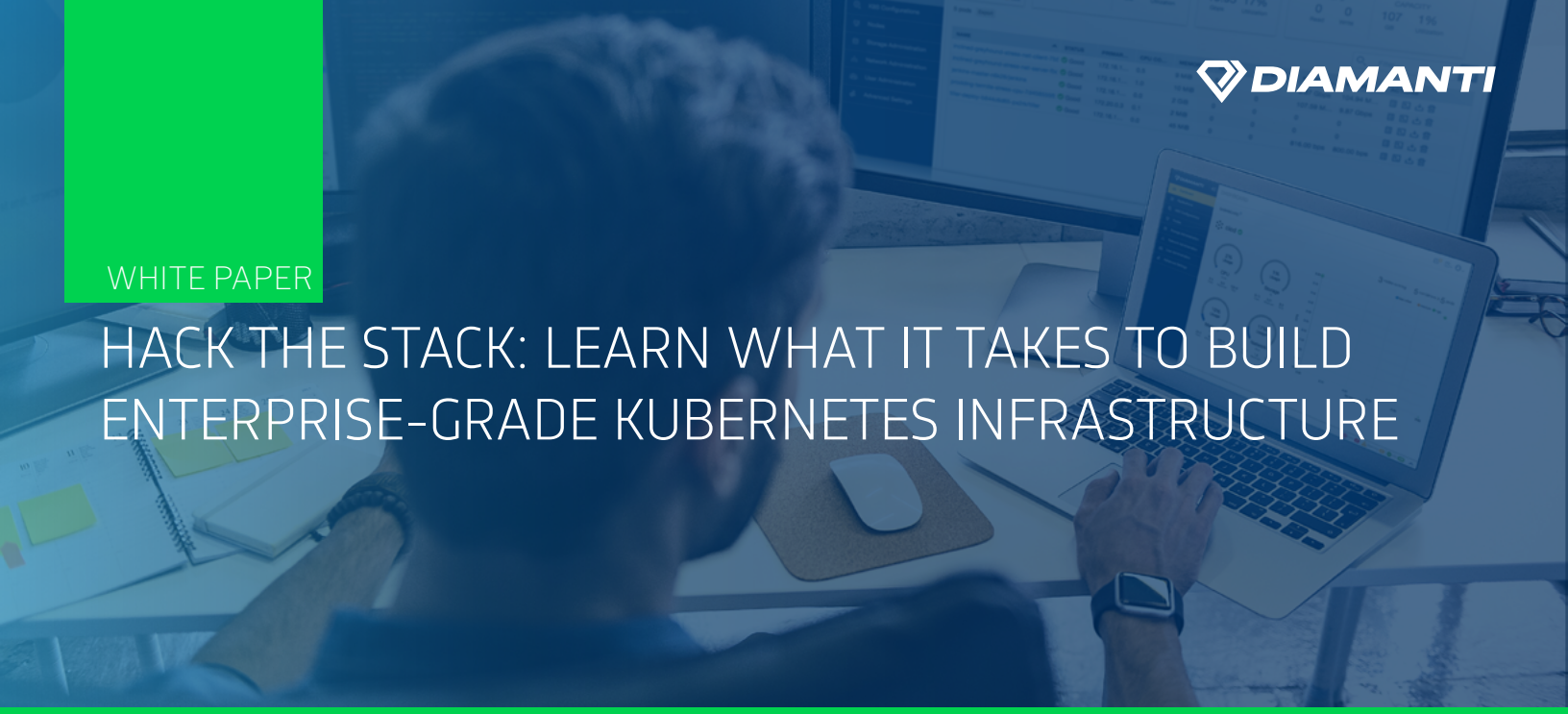

DIAMANTI

WHITE PAPER

HACK THE STACK: LEARN WHAT IT TAKES TO BUILD ENTERPRISE-GRADE KUBERNETES INFRASTRUCTURE

SYNOPSIS

Many organizations that have successfully operationalized cloud-native applications then turn to modernizing legacy applications, with the goal of building their own Kubernetes environments on premises. But containers use infrastructure resources differently than legacy applications, so users face significant challenges configuring, managing, and scaling infrastructure for Kubernetes.

This paper examines Kubernetes requirements for storage, networking, and a host of other infrastructure elements to give IT operations professionals actionable guidance when planning on-premises Kubernetes environments.

INTRODUCTION

Enterprises charting a path toward widespread digital transformation will likely face major challenges deploying Kubernetes on premises and running existing legacy applications in containers. The first burning question you will need to answer is: what do I need from an infrastructure perspective?

IT operations teams are often compelled to “roll their own” Kubernetes stacks using the infrastructure they currently have, but they can run into significant infrastructure challenges in the process. To understand your Kubernetes infrastructure requirements,

it's helpful to first examine how modern containerized applications differ from traditional, monolithic applications.

Most traditional applications combine frontend, backend, and business logic components into one big monolith. Consequently, such monoliths are typically tied to static IP addresses and operating systems, with underlying static storage for application data. As a result, it is impossible to scale, upgrade, or replace a single application component independent of the other components.

Cloud native applications are made up of many smaller, composable pieces. The application is designed as a set of microservices, where each distinct service is a self-contained unit comprised of application code, libraries, runtime, etc. and is independent of other services. Microservices are not tied to a static IP address or static storage. This provides the flexibility to scale, upgrade, or replace a service with no impact to other services. It also requires a new approach to networking and persistent storage; this is where many IT operations professionals spend the bulk of their time as they roll out Kubernetes clusters.

KEY ELEMENTS AND CAPABILITIES OF ON-PREMISES KUBERNETES INFRASTRUCTURE

The following are the key infrastructure components and features required to support a successful Kubernetes deployment on premises:

STORAGE
- Automated provisioning
- Data high availability (HA) and agility

NETWORKING
- Ingress control

OTHER SERVICES / CAPABILITIES
- Quality of Service (QoS)
- Unified multi-tenancy
- Consolidated quota management
- Disaster recovery

THE IMPORTANCE OF INFRASTRUCTURE AUTOMATION

Infrastructure automation is one of the critical elements of a Kubernetes environment. Along with orchestrating application containers, Kubernetes has the ability to integrate tightly with backend infrastructure and automate the entire application deployment process.

Cloud environments automate infrastructure for customers. But how do you achieve the same level of automation in your data center, especially when you're running infrastructure products from more than one vendor? Below, we look at the role of automation for the key infrastructure elements.

STORAGE

AUTOMATED PROVISIONING

Automated storage provisioning allows storage to be dynamically created or deleted based on application demands. Without it, application developers have to explicitly request that a storage admin create storage volumes for them prior to deploying an application. Any new instance of the same application will also require an admin to allocate new storage. This manual storage provisioning is onerous, results in inevitable delays, and hampers application scalability.

With dynamic provisioning, new storage is automatically created or deleted when a new application (or a new instance of the same application) is run. The storage admin is no longer a bottleneck to scalability; their involvement is limited to the initial setup of storage quotas.

Applications such as Redis, Kafka, Postgres, and MongoDB take full advantage of this flexibility and scale automatically to meet user demand.

DATA HIGH AVAILABILITY AND AGILITY

Often, applications are designed to move across nodes—or from one data center to another—in response to hardware or network failures or as a result of changing user demands. It makes sense that application data should move along with the application instance, or the data should be made available to the new node or data center hosting the application instance.

To achieve application HA, data needs to be stored redundantly in more than one node or data center. The underlying storage subsystem should provide this ability for applications, so applications can simply access the same data when they move.

It is not possible to make application data highly available across all nodes—or all

data centers—all of the time. Infrastructure should be capable of integrating with the orchestration systems that schedule applications and affinitize applications to nodes and data centers where the data they need is available.

Kubernetes integrates well with single-vendor cloud environments but can have difficulty operating across heterogenous or multi-vendor enterprise environments. In a multi-vendor environment, storage, networking, identity management and load balancing come from different vendors and each of them have their own plugins and dependencies to integrate with Kubernetes and do not work in cohesion like in single-vendor cloud environments. Also, day two problems like upgrades are tied to multiple vendors and there is no guarantee all of them work together. Therefore, it is recommended to use enterprise Kubernetes offerings which are built specifically for heterogeneous environments.

NETWORKING

There is no standard way of networking pods in Kubernetes. While Kubernetes—as an open source platform—allows for innovative new approaches to networking, it also makes it difficult to cleanly integrate with external hardware and software without a defined standard.

There are various open-source networking solutions available for Kubernetes. With the majority of these solutions, IP address management is completely hidden and has to be managed using VXLAN overlays. This works fine if all your services are running within the Kubernetes environment, however, it is not possible to expose those services to other external services without the use of Network Address Translation (NAT). Furthermore, the translated IP address will not be made highly available. In this scenario, there are multiple drawbacks to consider:

PERFORMANCE DEGRADATION
With overlays and an IP model based on NAT, traffic must always pass through the host network namespace and software switching layer.

INABILITY TO DISTRIBUTE TRAFFIC ACROSS DIFFERENT 10G INTERFACES
Most Kubernetes networking solutions assume one 10G interface and do not support more than one 10G interface.

INABILITY TO SUPPORT MORE THAN ONE IP ENDPOINT PER CONTAINER
Most Kubernetes networking solutions do not support more than one IP endpoint. This makes it impossible to support applications which require more than one IP endpoint per container.

LACK OF ISOLATION
Any traffic that passes through the host network namespace won't be isolated and is therefore not secure and can be snooped.

NO HA FOR NETWORKING
Networking is not highly available as only 10G interfaces are supported. It is possible to get around this via bonding, but not many network solutions use bonding as the default deployment model.

NO MULTI-TENANCY
Most available networking solutions do not offer multi-tenancy or network microsegmentation for applications.

The above drawbacks can be mitigated, but a lot of work is required in order to come up with a homegrown deployment methodology.

Few networking solutions support public and private IP endpoints while also enabling high availability, isolation, and microsegmentation. We recommend that IT teams decide which of these features are most essential and choose the best networking solution keeping in mind that some trade-offs may be required when choosing solutions that will run on existing infrastructure.

INGRESS

Applications running on Kubernetes are exposed to the outside world via Kubernetes Ingress, which is suited for Layer-7 applications. However, enterprise organizations will likely have external load balancers already deployed, therefore there is an easy migration path from hardware to software-based ingress control.

QUALITY OF SERVICE (QOS)

One of the biggest problems with running microservices or cloud native applications is noisy neighbor effects. How can the application be shielded? How can application service levels be guaranteed when running multiple application containers on the same node? Typically, CPU and memory resources can be managed by using the default controls provided by the underlying operating system. However, storage IOPS and network bandwidth cannot be managed unless the underlying infrastructure provides dedicated controls.

Infrastructure should be able to guarantee storage IOPS and network bandwidth service levels in Kubernetes environments based on application priority. However, not many infrastructure providers offer this capability. Delivering bona fide QoS for application containers requires tight integration with Kubernetes using its plugin and API model and the ability to separate out storage and network traffic.

MULTI-TENANCY

In public cloud environments, multi-tenancy is built in across the different layers of infrastructure. In enterprise container environments, the same degree of multi-tenancy isn't achievable with infrastructure solutions sourced from multiple different vendors. Hyperconverged infrastructure (HCI) solutions, which integrate network, storage, and compute in the same platform, can make it much simpler to manage multi-tenancy. Tight integration with the orchestrator is also important.

CONSOLIDATED QUOTA MANAGEMENT

Another big challenge is consolidated quota management, which is a must for proper capacity planning and billing for different projects across the enterprise. There are existing standalone tools that can be leveraged, but most of them involve manual processes. How is it possible to automate quota enforcement and manage application resource utilization? At the same time how do you get consolidated quota management across different layers of infrastructure? Again, full integration of the underlying infrastructure with Kubernetes using the Kubernetes plugin model and API is required to achieve this.

DISASTER RECOVERY

Enterprises use backup and data replication tools to achieve disaster recovery. Most of the backup and replication vendors do not currently support Kubernetes environments, but the lack of dedicated solutions can be mitigated by the following practices:

BACKUP

- Deploy out-of-band control software to snapshot data and backup to third-party software
- Use application-level backup tools like SQL Dump, dump the data to an external NFS volume, and backup that NFS volume

REPLICATION

- Use the storage provider's replication feature
- Use helper tools like rsync to replicate data to another Kubernetes cluster

CONCLUSION

Successful on-premises Kubernetes deployments require careful attention to a number of key infrastructure considerations. Storage must support automated provisioning and address your HA and DR needs. Networking must be extensible to support external services running outside of the Kubernetes environment in order to support legacy applications, and you may need additional capabilities such as QoS and quota management. It can be complicated to architect a solution that satisfies your requirements and supports both cloud native applications and containerized legacy applications on top of existing heterogeneous infrastructure. Many teams find themselves dedicating a lot of time and energy but end up with a solution that falls short of their needs in key ways.

Diamanti solves these challenges with the only turnkey solution for Kubernetes on the market today. Our enterprise-class, bare-metal container platform provides high-performance compute, plug-and-play networking, persistent storage, Docker, and Kubernetes—integrated in a simple solution with full-stack support.

Efficient, hyperconverged infrastructure provides high-performance storage and networking that integrates with your existing data center technologies. The result is a highly-available pool of CPU, memory, network, and storage resources delivered to containers on-demand, with full QoS for all resources including storage and networking, something no other vendor offers.

Diamanti integrates all the hardware and software you need, so it can be fully deployed and operational in minutes. You'll be able to start running microservices applications and containerized legacy applications immediately, without having to spend weeks or months on a do-it-yourself solution.

ABOUT DIAMANTI

Diamanti's D10 bare-metal container platform gives infrastructure architects, IT operations, and application owners the speed, simplicity, efficiency, and control they need to run stateful containerized applications at scale. With open-source Docker and Kubernetes fully integrated, together with purpose-built hardware and complete support for the entire stack, the Diamanti D10 is a proven full container stack that deploys in minutes.

For more information about Diamanti's bare-metal container platform, please visit www.diamanti.com/product

DIAMANTI

www.diamanti.com | info@diamanti.com | @DiamantiCom

Diamanti, Inc. 111 N Market St., Suite 800, San Jose, CA 95113

(408) 645-5111