e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:01/January-2023 Impact Factor- 6.752 www.irjmets.com

HIGH AVAILABILITY AND DISASTER RECOVERY IN AZURE FOR KUBERNETES

Avinash Ganne[*1]

[*1]Sr. SAP Basis Cloud Architect, Raley's, Sacramento, California, USA.

DOI : https://www.doi.org/10.56726/IRJMETS32765

ABSTRACT

Azure Kubernetes Service (AKS) simplifies deploying a managed Kubernetes cluster in Azure by offloading the operational overhead to Azure. AKS is an orchestration tool which means that a complex application made up of multiple containers Kubernetes tool will help achieve high availability, scalability and disaster recovery application setup. High availability and scalability have worker nodes of kubernetes cluster server holding a replica of application and a database application and also have an ingress component which basically handles every incoming request to application. If someone accessed app website on a browser the request would come in to ingress an ingress component will actually be their own parts on each server are replicated as well. When a user visits the app website on browser a request is made and handled by ingress first and ingress is load-balanced have replicas of ingress on multiple servers ingress will then forward that request to a service for application and service is a load balancer as well then direct that request to the respective replicas of the ports for that request. It must be noted that application design should also support replication and request handling because these are just tools that Kubernetes offers to make properly designed application highly available in highly scalable Kubernetes-based infrastructure with high availability leveraging Azure Stack Hub's Azure Kubernetes Service (AKS) Engine. This situation is typical for businesses that handle crucial tasks in environments that are tightly controlled and regulated companies and industries including banking, defense, and government.

Keywords: Azure Kubernetes Service, Disaster Recovery, Cloud Computing.

I. INTRODUCTION

In cloud platform's disaster recovery (DR) capabilities should be a top priority[1-3]. Utilizing a backup data center or region in the cloud for disaster recovery is frequently the first step in cloud adoption. Cross-region backup and restoration for Azure Kubernetes Service or disaster recovery (AKS) Cloud-native, highly accessible, robust, and easy to construct are all characteristics of persistent volume storage.

If the server is completely crashed, and all the parts that are running on it die user would still have replicas of an application running; there will be no downtime. In the meantime, a Kubernetes master process called controller manager would actually schedule new replicas of the died pods on another worker node. Server will recover the previous load-balanced and replicated application state which means while the node servers actually do the work of running the applications master processes on the master nodes. The master node actually monitor the cluster state and make sure that if a pod dies, it automatically gets restarted or if something crashes in the cluster that it automatically gets recovered. An important master component that is used to manage the cluster mechanism to run properly is the store which stores the cluster state like the resources available on the notes and the pod state, etc. at any given time.[4]

For example, if a pod dies and its CD is updated about it, that's how the controller management would know that it should intervene and make sure the new pot gets restarted. When this happens again CD gets updated and because the store always has the current state of the cluster it's also a crucial component in disaster recovery of Kubernetes clustered applications and the way disaster recovery mechanism can be implemented Kubernetes is to create debt CD backups and store them into remote storage and this backup share in the form of 8 CD snapshots. Kubernetes doesn't manage or take care of baking updated CD snapshots on remote storage; this is the responsibility of the Kubernetes cluster administrator.[5] This storage could be completely outside the cluster on a different server or may be even cloud storage an important note is that HCD doesn't store database or application data that data is usually also stored on remote storage where the application thoughts actually have reference to the storage so that they can read and write the data from and this remote storage just like the HDD snapshots backup location isn't managed by kubernetes and again must be reliably backed up and stored outside of the cluster. This is usually how production clusters are setup a reliable backup and replication of HDD snapshot and application data is in place if the whole cluster were to crash including the worker nodes and the master nodes themselves. It would be possible to recover the cluster state on completely new machines with new worker nodes and also new master node using the HDD snapshot and the application data and of course user can even avoid any downtime between the cluster crash and a new cluster creation by keeping a backup cluster.[6]

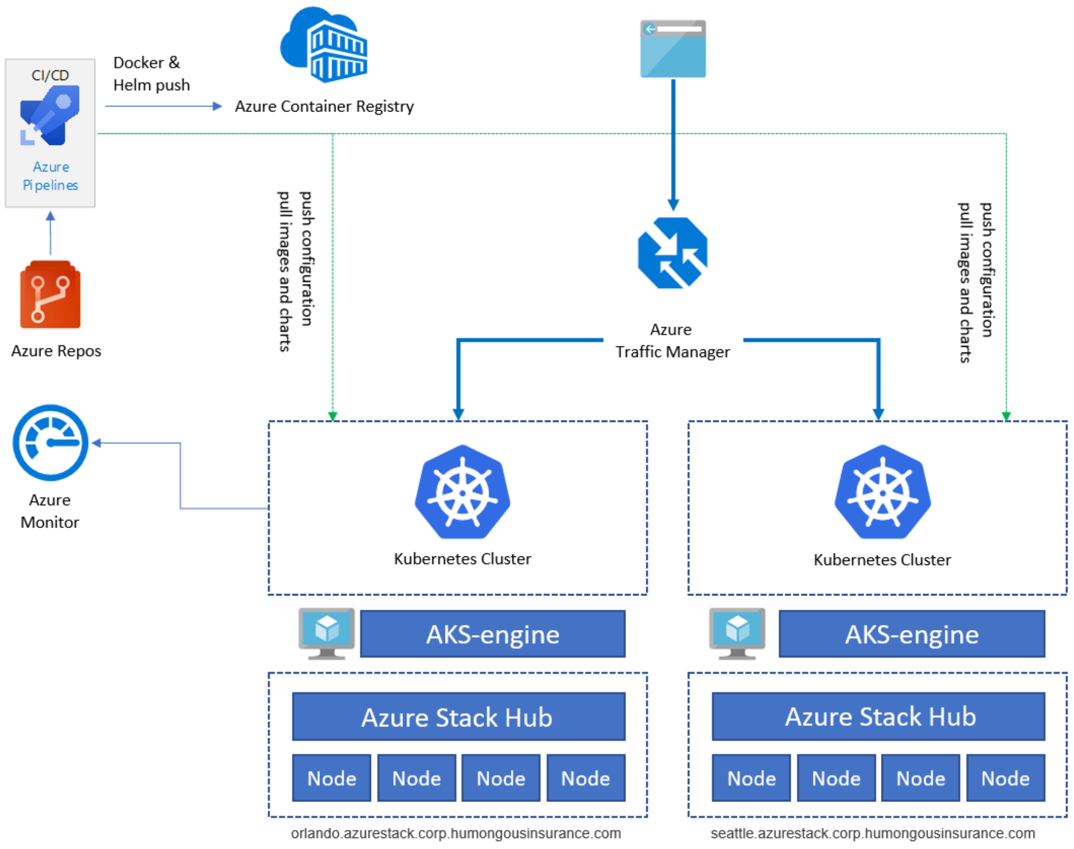

Figure 1: Asynchronous replication

To achieve high availability (HA), we need to run the deployment at least twice on two different Azure Stack Hub instances - they can run either in the same location or in two (or more) different sites. Services like Azure Container Registry, Azure Monitor, and others, are hosted outside Azure Stack Hub in Azure or on-premises. This hybrid design protects the solution against outage of a single Azure Stack Hub instance.

II. HIGH AVAILABILITY KUBERNETES CLUSTER PATTERN

The program's exposure to and accessibility from the Internet are the main factors to take into account for the application layer. In terms of Kubernetes, exposing a deployment or pod via an Ingress Controller or a Kubernetes Service is Internet accessibility. An application is not automatically made available through the Internet just because it is exposed using a public IP address by a load balancer or ingress controller. Not all public IP addresses are actually Internet-facing, therefore it's feasible for Azure Stack Hub to have one that is only accessible from the local intranet.[4] Using transparent or non-transparent proxy servers may be necessary in some business situations. For various parts of our cluster, some machines need particular configuration.[5] Instances of Azure Stack Hub must communicate with each other across clusters. Various Kubernetes clusters are running on various Azure Stack Hub instances in the test deployment. They are connected by "external traffic," which includes traffic involved in duplication across database. To connect Kubernetes on two Azure Stack Hub instances, external communication must be routed over either a Site-to-Site VPN or an Azure Stack Hub Public IP address. The Kubernetes cluster doesn't always have to be reachable online. When using kubectl, for instance, the Kubernetes API is utilized to manage a cluster. Anyone who manages the cluster or deploys software and services on top of it has to have access to the Kubernetes API endpoint.[6] Plan out the ultimate networking configuration before building a Kubernetes cluster utilizing AKS Engine. It might be more effective to deploy a cluster into an existing network rather than building a separate Virtual Network. Using an existing Site-to-Site VPN connection that is already set up in your Azure Stack Hub environment, for instance. Accessing the Azure Stack Hub management endpoints is referred to as infrastructure. The Azure Resource Manager admin and tenant endpoints, as well as the tenant and admin portals, are examples of endpoints. These endpoints are necessary for Azure Stack Hub's basic functionality.[7] It can immediately take over when the active cluster or the current cluster crashes or dies. There is one thing that should be noted here users can actually achieve this setup with load balancers and replicas also without Kubernetes. For example, on AWS instances using the AWS load balancer, etc., however, there are a couple of advantages that users have with Kubernetes that users don't have with other tools.

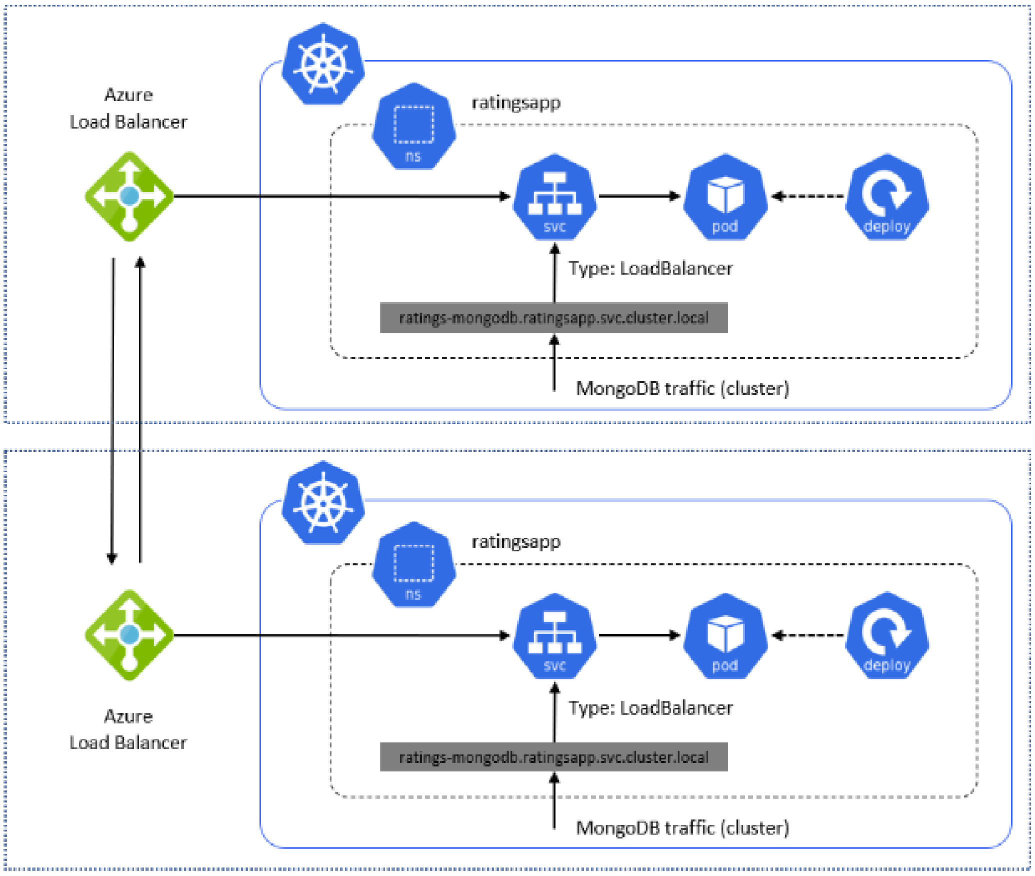
Azure
Load Balancer
ratingsapp
ns
svc
pod
deploy
Type: LoadBalancer
ratings-mongodb.ratingsapp.svc.cluster.local
MongoDB traffic (cluster)
ratingsapp
ns
svc
pod
deploy
Type: LoadBalancer
ratings-mongodb.ratingsapp.svc.cluster.local
MongoDB traffic (cluster)
Azure
Load Balancer

Figure 2: Azure Load Balancer

If a user creates this setup one is the replication is made much easier using Kubernetes; the only thing that the user has to do is just declare how many replicas of a certain application beat the user's own application or a database application needs and the Kubernetes component takes care of actually replicating its second one is the Kubernetes self-healing feature.[7] It basically means that if a pod dies, there should be a process that monitors the state that detects that a replica died and automatically restarts a new one, and again users have this feature out of the box from Kubernetes. The smart scheduling feature of Kubernetes means that, for example, if the user has 50 worker servers and application containers will run on with Kubernetes user doesn't need to decide where to run the container just need a new replica of a pod. Kubernetes smart scheduler basically goes and finds the best fitting work node among those 50 worker nodes maybe to schedule container by comparing how much resources a worker node has available or free, and overall many features that user could also do elsewhere on other platforms like AWS is made simpler or it's easier to create and configure in Kubernetes like service as a load balancer.[8] For high availability that user really targeting is continued availability with resilience to more localized failures whereas disaster recovery that's really looking at having that business continuity in the case of a more widespread disaster and being able to quickly recover from a

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:01/January-2023 Impact Factor- 6.752 www.irjmets.com

large disaster scenario. So with high availability that's really within a region it's more localized issue and failover happens within the magnitude of a second it's really being able to quickly recover and user have his data back up and running whereas disaster recovery this spans a larger scale of issues where user failover would actually happen across different azure regions and this would really target more of that large scale type of disruption than high availability which focuses on an within region localized issue. For high availability user really for SQL DB target synchronous replication that way user always have a replica that's fully caught up with primary and user have that and just have caught up data which is there for localized failover and then for disaster recovery use asynchronous replication since this is targeting more across regions user don't have to worry about having all of waiting for all these commits to happen to a replica that's in another region so that is more focusing on the performance aspect.[9]

III. ASYNCHRONOUS REPLICATION FOR DISASTER RECOVERY

Use asynchronous replication for disaster recovery, and then in terms of SLA for high availability, how to measure this in SLA is something called uptime percentage, which is the number of nines that users typically see. If the user hears something like we have four nines, that is referring to uptime percentage, which is a high availability SLA, so the amount of time your database is up and running. Whereas for disaster recovery, the SLA metrics that are used are RPO recovery point objective and RTO recovery time objective.[10] Starting with the general purpose tier what this architecture looks like user have primary computer replica which connects to remote storage which uses locally redundant storage so that's replicated within same localized area within a data center within an azure region and then this also has backup files which are by default configured to be Geo-redundant. It provide capabilities you to change this backup configuration to be zone redundant or local redundant and then how failover happens your database would need to failover because of some kind of issue there would be nodes of spare capacity that database can fail primary compute replica can fail over to within the same data center within azure region. This is essentially how a general purpose database looks whenever create in azure SQL database just have remote storage connected to compute replica then for business critical tier how this differs from general purpose is actually have four replicas total have primary replica and then it's connected to three additional secondary replicas based on always-on availability groups which may be familiar with from SQL server. It's the same concept except this is configured out of the box for whenever create a business critical azure SQL database [11]to have four total replicas connected through always on availability groups each of replicas uses the local storage for any storage both the compute and storage are connected locally and it's just the backups that use remote storage for and it's the same as with the general purpose tier how by default this is global redundancy backup storage however this can be configured to be zone redundant or local redundant. With these four replicas happens, in the case of an issue with the primary replica, user can failover to a secondary replica since user already have four created; another capability of the business critical tier is since you have each of these replicas one of the secondary's can actually be used to send read-only workloads for reading scale out it can configure business critical database to have read-write workload sent to primary and then read only once sent to the secondary as well so not only does it provide that failover capability but the user also have read scale-out capability out of the box with a business-critical Azure SQL database.[10]

IV. CONCLUSION

The most crucial factor for the application layer is whether the application is exposed to and reachable over the Internet. From a Kubernetes context, exposing a deployment or pod via a Kubernetes Service or an Ingress Controller constitutes Internet accessibility. The use of a Load Balancer or an Ingress Controller to expose an application using a public IP does not automatically make the application available via the Internet. Not all public IPs are really Internet-facing; hence it is conceivable for Azure Stack Hub to have a public IP address that is only accessible on the local intranet. Natural disasters like hurricanes, earthquakes or even a man-made disaster like a war could cause permanent damage or partial loss in an entire data center; the recommendation is that every production database should be using strategy Azure SQL. It is not a managed service to deploy Kubernetes on Azure Stack Hub using AKS Engine. Using Azure Infrastructure-as-a-Service, an automated Kubernetes cluster deployment and configuration is being performed (IaaS). It offers the same level of availability as the underlying infrastructure as a result.

V. REFERENCES

[1] B. Weissman and A. E. Nocentino, "A Kubernetes Primer," in Azure Arc-enabled Data Services Revealed: Springer, 2022, pp. 1-23.

[2] S. Buchanan and J. Joyner, "Azure Arc-Enabled Kubernetes and Servers."

[3] S. Buchanan, J. Rangama, and N. Bellavance, "Operating Azure Kubernetes Service," in Introducing Azure Kubernetes Service: Springer, 2020, pp. 101-149.

[4] A. Ganne, " Cloud Data Security Methods: Kubernetes vs Docker Swarm."

[5] K. K. Sethy, "Serverless implementation of Data Wizard application using Azure Kubernetes Service."

[6] K. Malathi, "Managing the Stateful Applications for High Availability using Novel Kubernetes based Microservice Architecture over Cloud based Azure Virtualization Architecture," Journal of Pharmaceutical Negative Results, pp. 1536-1547, 2022.

[7] K. K. Sethy, D. Singh, A. K. Biswal, and S. Sahoo, "Serverless Implementation of Data Wizard Application using Azure Kubernetes Service and Docker," in 2022 1st IEEE International Conference on Industrial Electronics: Developments & Applications (ICIDeA), 2022: IEEE, pp. 214-219.

[8] A. E. Nocentino and B. Weissman, "Azure Arc–Enabled Data Services and High Availability for SQL Server on Kubernetes," in SQL Server on Kubernetes: Springer, 2021, pp. 197-214.

[9] G. Sayfan, Mastering kubernetes. Packt Publishing Ltd, 2017.

[10] B. Familiar and J. Barnes, "Sensors, Devices, and Gateways," in Business in Real-Time Using Azure IoT and Cortana Intelligence Suite: Springer, 2017, pp. 127-168.