Federated Multi-Cloud Kubernetes Deployment Using Infrastructure As Code

Harsh Dubey hd2225@nyu.edu
Courant Institute of Mathematical Sciences
New York University
New York, NY, USA

Hemant Ramawat hr2378@nyu.edu
Courant Institute of Mathematical Sciences
New York University
New York, NY, USA

Adidev Vinaik av3034@nyu.edu
Courant Institute of Mathematical Sciences
New York University
New York, NY, USA

Abstract

Multi-cloud applications are seen to be performing much better than applications using single-cloud deployment strategies. Kubernetes has become one of the nuts and bolts of the cloud infrastructure that we see today. But how about we could leverage Kubernetes from multiple clouds into our application? Although this question seems difficult, it has become relatively easy to answer and implement given the advances in the field of cloud, its scope, and the tools around it. In this project, we will use a multi-cloud Kubernetes federation to deploy our machine learning application across clouds. We will start by exploring the concepts and knowledge needed to achieve federation. Then we will look at the architecture and implementation that makes it possible. Finally, we will do some analysis on how and where such architecture can be useful.

Keywords: Cloud Federation, Kubernetes, Consul, IAC

Datasets: MNIST

Code Repository: federated-multicloud-kubernetes-git-repo

1. Introduction

A cluster of Kubernetes nodes can be deployed and managed across many cloud environments, such as Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure, using a federated multi-cloud Kubernetes deployment (Mul). This method gives you more flexibility and control over your deployment while letting you take use of the advantages of various cloud platforms.

Increased availability, scalability, and cost effectiveness are just a few advantages that federated multi-cloud Kubernetes installations may offer (Dhanapal, 2022). It's crucial to carefully assess whether this method is the appropriate fit for your needs, but they can also be difficult to set up and manage.

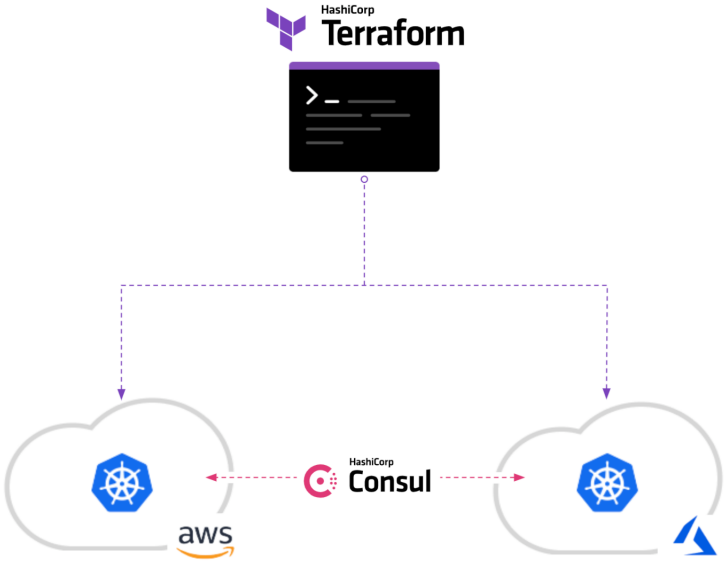

Figure 1: Federation of Kubernetes across clouds

2. Background

In this section, we will deep dive into understanding, what led to what we see today as a new era in the field of cloud strategies and adoption.

2.1 Motivation

- For container orchestration, Kubernetes has gained popularity and is available in a variety of configurations (Kub, a), including as a managed service on various cloud platforms (such as Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Service (EKS), Oracle Kubernetes Engine (OKE), and Azure Kubernetes Service (AKS)) and in self-managed versions like DIY and Konvoy.

- The proliferation of Kubernetes has fueled in part the growth of multi-cloud automation infrastructure. Due to enterprises' need to automate and streamline the deployment and maintenance of their applications and services, the development of cloud automation infrastructure has played a part in the spread of Kubernetes. Since it makes it simple for businesses to build and handle containerized apps at scale, Kubernetes has established itself as a fundamental component of many cloud systems (Perry).

- The hype around multi-cloud has also contributed to its widespread adoption (Clo, 2022). A multi-cloud strategy gives businesses more flexibility and control over their deployment while letting them benefit from the advantages of several cloud platforms. By offering a consistent platform for delivering and managing applications across various cloud environments, Kubernetes can be crucial in allowing a multi-cloud approach.

- Multi-cloud's dependability and high availability have also contributed to its popularity. Applications can scale to meet demand thanks to infrastructures with self-healing, auto-scaling, and rolling update features, which help to assure their availability at all times. In the modern digital environment, where consumers expect smooth, always-on access to applications and services, this is especially crucial.
- The rapid adoption of multi-cloud has also been aided by the growth in global user numbers. A highly accessible and scalable infrastructure is becoming more and more necessary as consumers from all over the world access apps and services from a range of devices and places. As it enables businesses to easily deploy and manage applications across numerous locations and cloud environments, multi-cloud is well suited to address these objectives (Ter, b).

2.2 The Hype of Multi-Cloud

There are a number of examples of the hype around multi-cloud adoption:

- Large tech businesses are embracing multi-cloud strategies: To profit from the advantages of utilizing several cloud platforms, many major tech companies, like Netflix, Airbnb, and Twitter, have embraced multi-cloud strategies. These firms have employed a range of strategies, such as the usage of several public cloud platforms, hybrid cloud architectures, and self-managed private clouds (Bhandari).
- Multi-cloud adoption is expected to rise rapidly in the upcoming years, according to researchers. Several research organizations and analysts have made this prediction. For instance, a survey by Gartner predicted that, up from less than 50% in 2018, 90% of enterprises would adopt a multi-cloud or hybrid cloud strategy by 2022.
- At industry events, multi-cloud techniques are frequently discussed: The advantages and difficulties of implementing this strategy have been the focus of several presentations and panels at industry conferences and events (Srinivas and Kuo; Kub, b).
- Vendors are advertising multi-cloud solutions as a method for businesses to profit from the advantages of utilizing several cloud platforms. These vendors include cloud providers and third-party software providers. These solutions could consist of services for application migration between clouds as well as tools for managing and deploying apps across various cloud environments.

3. Context and Related Work

Monolith applications:

- All components packaged and deployed as a single application.
- Bug fixes mean deploying all components as a group.

Microservice applications:

- Discrete packaging and deployment of different components in an application.
- Redeploying individual components as and when needed ensures development agility.

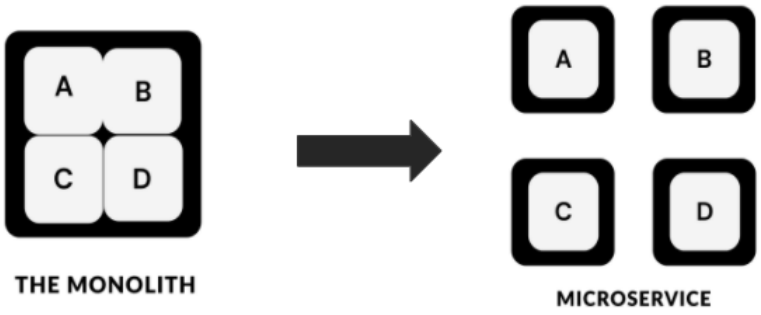

Figure 2: From Monolith to Microservice

3.1 The Problem of Discovery, Configuration, and Segmentation (DCS)

3.1.1 Discovery

- Front each service with a load balancer to enable discovery.
- Proliferation or explosion of Load Balancers.
- Load Balancer becomes the single point of failure.
- **Solution: Central Registry**

3.1.2 Configuration

- No consistent view of the application configuration.
- Decentralization of configuration can lead to a non-unified, ambiguous view of the other edges of the application.
- **Solution: Centralization of configuration**

3.1.3 Segmentation

- No defined segmentation of the network making the traffic pattern complicated.
- The communication pattern between services becomes a challenge given no network pattern restrictions.
- **Solution: Service Graph**

4. Service Mesh: The Resolution of DCS

A service mesh is a software architecture pattern that allows you to manage and monitor the interactions between microservices in a distributed application. The stability, security, and scalability of the application can all be enhanced by giving the microservices a uniform approach to govern and control their communication(Und).

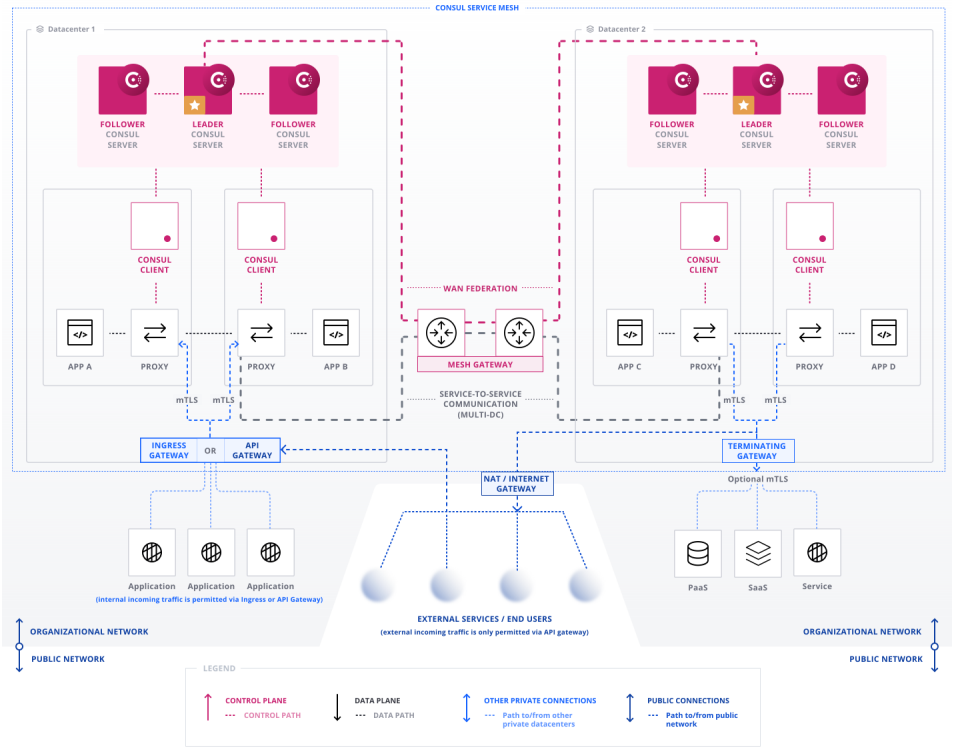

Figure 3: Service Mesh Across Clusters

Along with the microservices, a set of sidecar proxy containers are often deployed to form a service mesh. With functions like load balancing, service discovery, request routing, and monitoring, these proxies manage communication between the microservices.

Due to the fact that they offer a central point of control and management for the communication between the microservices, service meshes can be especially helpful in complicated, distributed applications that are made up of numerous microservices. As they give a variety of information and insights on the functionality and behavior of the microservices, they can also aid in enhancing the observability of the application.

4.1 Consul

Consul is a tool that helps you manage and secure the services that are deployed across your network (Con, a). The control plane, which is part of it, has a central registry that records the services' IP addresses. On clusters of nodes, such as servers, cloud instances, virtual machines, or containers, the control plane operates as a distributed system. Consul uses proxies, which are a component of the data plane, to communicate with the services that are a part of the network infrastructure. The processing of data requests and management of service-to-service communication is under the purview of the data plane. The core Consul workflow consists of the following stages(Con, b):

- Consul is a tool that assists with network service management and security. Without requiring human modifications to application code or fixed IP addresses, it includes

a central registry called the Consul catalog that enables services to automatically discover one another.

- Services can be manually registered with Consul or registered automatically using software like Kubernetes service sync. Services can also incorporate health checks to check for problems. You can locate healthy services in the Consul catalog using identity-based DNS from Consul, and you can manage the data flow over your network by setting up identity-based policies.
- Consul service mesh may also enable or deny access based on service identities, regardless of variations in compute environments and runtimes, and it offers security for microservice architectures with mTLS.

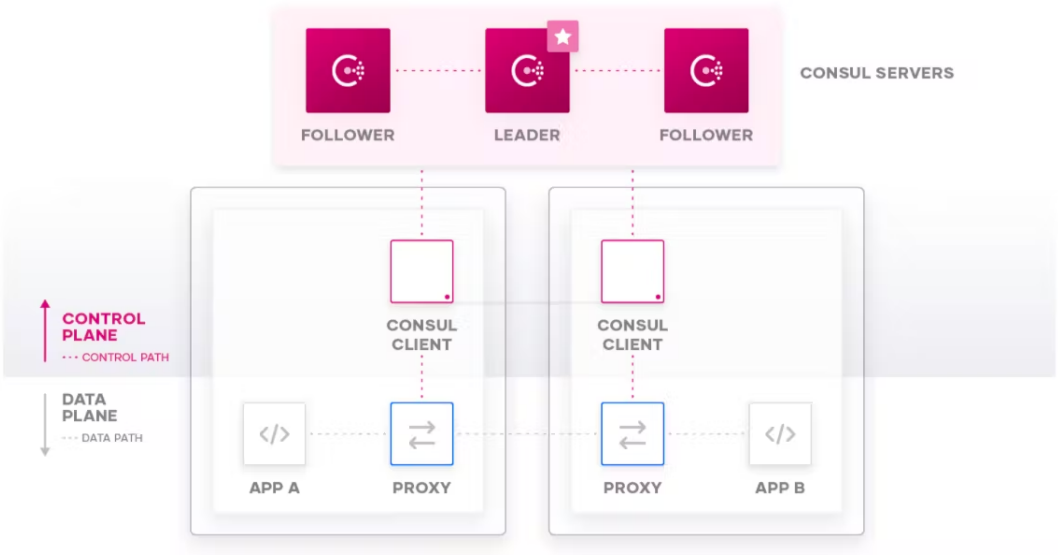

Figure 4: Consul as Service Mesh

4.1.1 Service Discovery

Through process of service discovery, services in a distributed system can find and connect with one another. The capacity of Consul clients to find and connect to services that have been registered with the Consul server is referred to as service discovery.

Consul offers service discovery using a mix of DNS and HTTP APIs. Clients can query the Consul server via the HTTP API to learn more about the services that are offered, or they can utilize the DNS server of Consul to search up the addresses of services by name.

The Consul architecture includes service discovery, which enables clients to find and contact the services they require in a fluid and distributed environment. As Consul clients can use the HTTP API to ask the Consul server for a service's status, it is also helpful for keeping track of the health of services.

Overall, service discovery in Consul helps to facilitate communication and resource sharing in distributed systems, making it easier for services to work together.

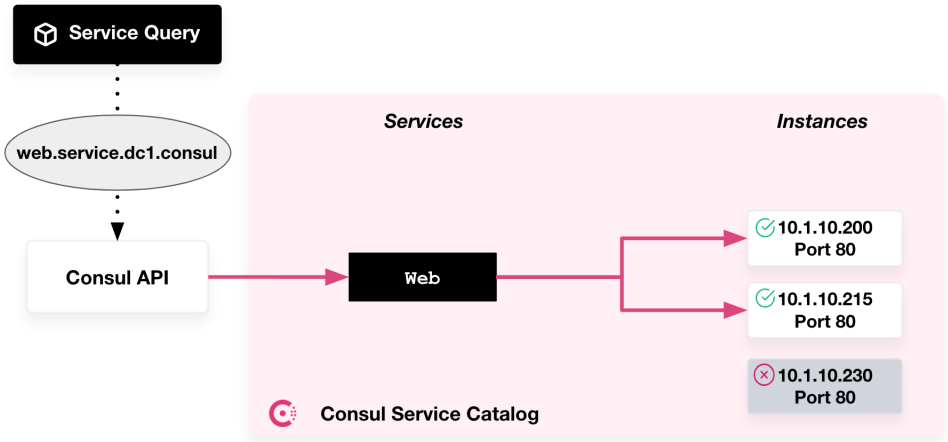

Figure 5: Service Discovery in Consul

Service discovery has several benefits for organizations:

- The ability to easily scale services, improve application resiliency, and simplify the process of communication between services.
- It can also help with load-balancing requests, reducing deployment times, and automating the registration and de-registration of services.
- Service discovery allows for dynamic discovery of IP addresses and ports, abstracts the discovery process away from applications, and uses health checks to ensure reliable communication between services.
- Service discovery can help to improve efficiency and effectiveness of distributed systems.

4.1.2 Gossip

The process of information exchange across nodes in a cluster is referred to as "gossip" in the context of the Consul service discovery and configuration tool. Consul employs the gossip protocol to communicate cluster status information to each cluster node (Gos, b).

Small messages, referred to as "gossip messages," are transmitted between cluster nodes as part of the gossip protocol. These messages include details about the cluster's condition, including the status of other nodes, the services that are accessible, and the condition of those services.

By using the gossip protocol, Consul can keep the cluster's nodes in sync and make sure that they are all aware of the same information on its current state. This is crucial for assuring the cluster's dependability and availability since it enables nodes to recognize changes inside the cluster and react appropriately (Vaughn).

Gossip is a crucial component of the Consul architecture and is used to make sure that every node in the cluster is aware of its status and can successfully communicate with other nodes (Gos, a).

Consul uses two types of gossip pools - one for the local area network (LAN) and another for the wide area network (WAN) - to perform different functions.

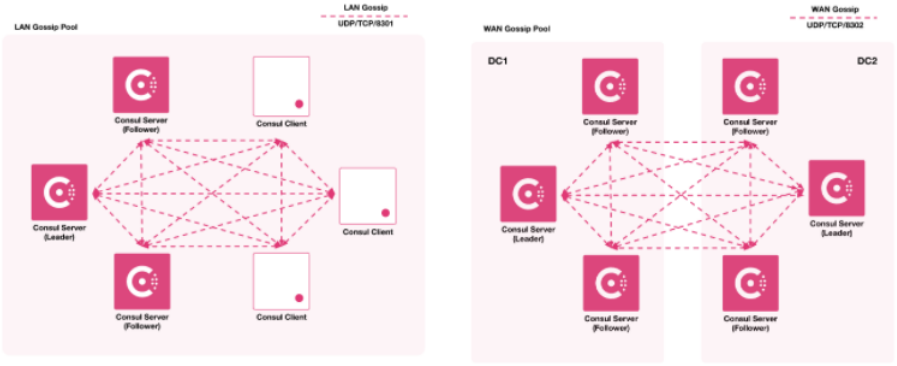

Figure 6: LAN Gossip in Consul

- **LAN Gossip**: The LAN gossip pool is used within a single data center and contains all members of the datacenter (both clients and servers). It allows clients to automatically discover servers and helps with failure detection and fast and reliable event broadcasts.
- **WAN Gossip**: The WAN gossip pool is unique to all servers and is used for cross-datacenter communication. It allows servers to perform requests across data centers and helps with failure detection in the event of a loss of connectivity in a data center or a single server in a remote data center.

Enabling WAN Federation Control Plane Traffic

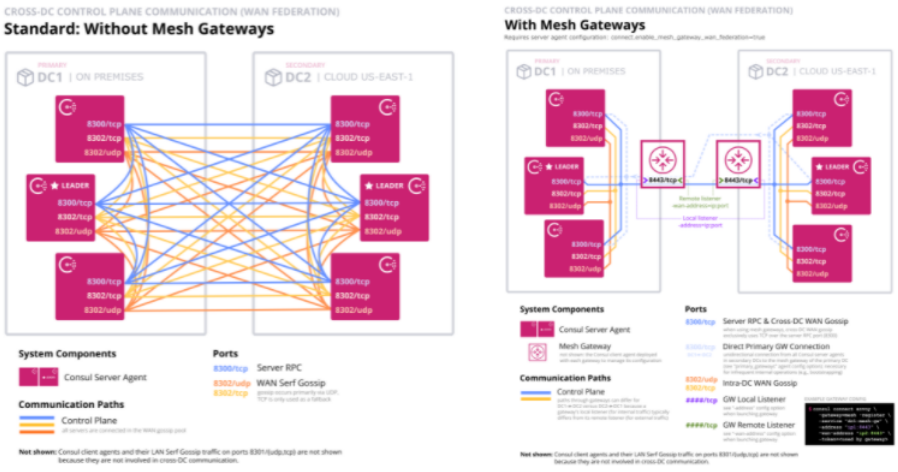

Figure 7: Mesh Gateways

5. Architecture

We have covered all the concepts needed to understand the architecture of federated cluster Kube deployment. In this section, we will deep dive into the architecture we are following in our application and its infrastructure.

Whole architectural setup can be understood can be understood and classified in to following steps:

- Provision an Amazon Elastic Kubernetes Service (EKS) Cluster using IAC.
- Provision an Azure Kubernetes Service (AKS) Cluster using IAC.
- Setup Consul Datacenters in both Kubernetes clusters using IAC.
- Deploy ProxyDefaults on the consul datacenters using IAC.
- Verify cluster federation by checking the deployments of pods and datacenters.
- Deploy Machine Learning Backend Service on AKS Cluster.
- Deploy Machine Learning Frontend Service on EKS Cluster.
- Finally port forward the deployment and access the application to verify working.

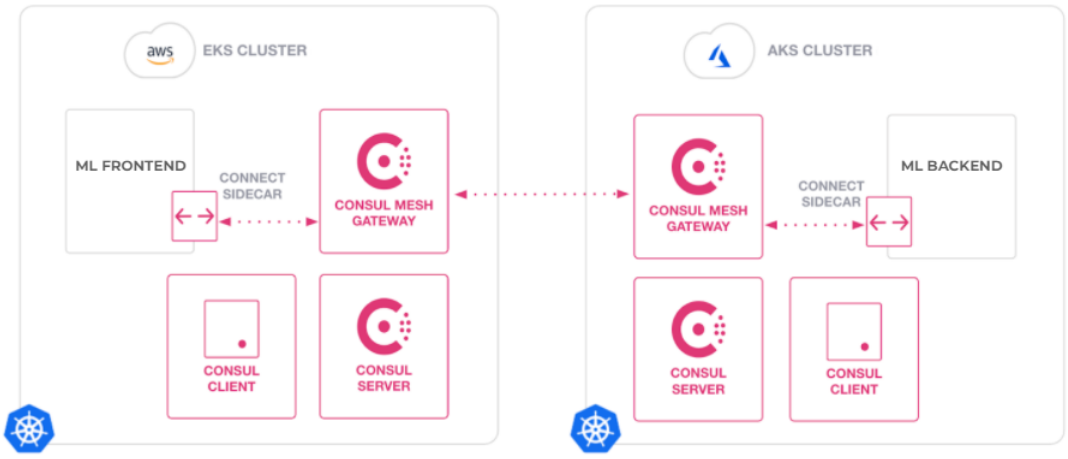

Figure 8: Architecture of Federated Deployment

6. Implementation and Deployment

To implement and deploy the federated multi-cloud Kubernetes with machine learning services (Ter, a) follow the steps below (Ter, c). Alternatively, one can also follow our project git repository which contains all the code, configurations and instructions from:
Code Repo: `federated-multicloud-kubernetes-git-repo`

- First create an AWS account and configure AWS CLI (Aws) -

  `Configure and Install AWS CLI`

- Next create an AKS account and configure AKS CLI (Azu) -

  `Configure and Install AKS CLI`

- Now, start by cloning the implemented code from our project repository -

```
git clone https://github.com/hardy30894/federated-multicloud-k8.git
```

- Now go in to the cloned repository as follows -

```
cd federated-multicloud-k8
```

- Remove previous datacenter configurations as follows:

```
rm -f ~/.kube/config
```

- Provision an EKS Cluster -

```
cd eks
terraform init
terraform apply --auto-approve
```

- Provision an AKS Cluster -

```
cd ../aks
terraform init
terraform apply --auto-approve
```

- Configure kubectl for both cloud providers -

```
cd ../eks
aws eks --region $(terraform output -raw region) update-kubeconfig --
  name $(terraform output -raw cluster_name) --alias eks
```

```
cd ../aks
az aks get-credentials --resource-group $(terraform output -raw
  resource_group_name) --name $(terraform output -raw
  kubernetes_cluster_name) --context aks
```

- Verify ProxyDefaults configuration is disabled -

```
cd ../consul
vi proxy_defaults.tf
# comment the ProxyDefaults configuration
```

- Deploy Consul and configure cluster federation -

```
terraform init
terraform apply --auto-approve
```

- Deploy ProxyDefaults -

```
vi proxy_defaults.tf
# Uncomment the ProxyDefaults configuration
terraform apply --auto-approve
```

- Verify cluster federation -

```
kubectl get pods --context eks
kubectl get proxydefaults --context eks
kubectl get pods --context aks
kubectl get proxydefaults --context aks
kubectl exec statefulset/consul-server --context aks -- consul
  members -wan
```

- Deploy the machine learning backend and frontend services -

```
cd ../counting-service
terraform init
terraform apply --auto-approve
```

- Run the application -

```
kubectl port-forward dashboard 9002:9002 --context eks
```

- Verify application is running -

```
http://localhost:9002/upload-image
```

7. Comparative Discussion

Node federation is something that we have been doing in our Kubernetes clusters for a good time now. By default, when we add more nodes to the Kubernetes cluster, we achieve node federation. Cluster Federation is what we are deploying in this experiment.

Figure 9: Node Federation vs Cluster Federation

Node federation, in the context of Kubernetes, refers to the action of joining several Kubernetes nodes to create a bigger, more potent cluster. A larger, more potent cluster that can handle more workloads and scale horizontally as the amount of data increases can be made by joining several nodes together.

Kubernetes cluster federation, on the other hand, refers to the process of connecting multiple Kubernetes clusters together to form a larger, more powerful cluster. A Kubernetes cluster is a group of nodes that are managed together and work together to run applications. By connecting multiple clusters together, it is possible to create a highly available, scalable cluster that can handle very large volumes of data and handle a high number of concurrent users.

On the other hand, Kubernetes cluster federation describes the procedure of joining several Kubernetes clusters together to create a bigger, more potent cluster. A group of nodes that are managed collectively and collaborate to run applications is known as a Kubernetes cluster. A highly available, scalable cluster that can handle very large volumes of data and a high number of concurrent users can be built by linking numerous clusters together.

8. Learnings

In this section, we will try to summarize our learnings by understanding what we achieved and what were the blockers in conducting the project.

8.1 Achievements

- Parsed how to provision resources dynamically and across clusters and providers using Infrastructure as Code.
- We understood cluster federation and utilized Helm charts to deploy Consul datacenters.
- We configured Consul Datacenters in both Kubernetes clusters i.e. AKS and AWS using IAC.
- Finally, we deployed the Machine Learning workload across two different federated datacenters.
- Given, that everything is stable, we verified cluster federation via kubectl and by port forwarding and accessing.

8.2 Blockers

- The support and documentation for multi-cloud deployments and federations are not extensive.
- The experiment/project is based on a relatively new and unfamiliar concept and hence required extensive study at every step.
- Given the limited availability of cloud federation management tools, there are still restrictions in terms of federation utilization.
- Also, given that this was a new concept to realize and work on, any modification required extensive changes which can be resolved with increasing interaction with the technology.

9. Future Scope and Improvement

- Due to the high demand for multi-cloud federation there is a need for easily and quickly deployable solutions.
- Some parts of the configuration still require manual intervention which raises the need for centrally managed services.
- The depth of configuration should be user-dependent. This means it should be easy for both new users as well as customizable for power users.
- Interfaces for interaction with the clouds could be simplified for the user. Added interoperability from the central dashboard with cloud ecosystems can be useful.
- There can be improvements in minimizing the latency across datacenter communication.
- There can also be improvements in authentication protocols and this would encourage more usage.

10. Conclusion

Multi-cloud adoption is the future. Although there are some blockers and limitations today, but with the pace at which we are seeing new tools and services around the multi-cloud federation, we can say that the benefits it presents are worth the adoption. Having the freedom to choose how to deploy workloads and not being tied to a specific vendor gives greater flexibility in terms of a multi-cloud strategy. Smooth failover and disaster recovery can reduce the risk of distributed denial-of-service (DDoS) attacks and single point of failure (SPoF) incidents. This approach allows for the use of all cloud services within any cloud ecosystem, and one benefit is the ability to geographically distribute apps and services. This can help organizations minimize latency and other performance issues by choosing a cloud provider with data centers located near their business users or customers.

11. Demonstration

We performed the implementation and deployment section of our end-to-end project, utilizing the architecture and tools as mentioned above. A video demonstrating the same can be viewed through below link:
Demo Video: `Federated Multi-Cloud K8 Deployment Using Infrastructure As Code`

Acknowledgments

We would like to thank Prof. I-Hsin Chung and Prof. Hao Yu for helping us to understand Cloud and its related terminology by guiding us in lectures and over slack to resolve the roadblocks. They were instrumental in providing us the opportunity to perform such a deep dive into these technologies and analyze the nascent and rich field of Federated Multi-Cloud Deployment. This project was carried out as a part of the curriculum for Cloud and Machine Learning course at NYU Courant Institute of Mathematical Science.

References

AWS CLI. https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html. Accessed: 2022-12-16.

Azure CLI. https://learn.microsoft.com/en-us/cli/azure/. Accessed: 2022-12-16.

Consul Architecture. https://developer.hashicorp.com/consul/docs/architecture, a. Accessed: 2022-12-16.

Consul on Kubernetes. https://developer.hashicorp.com/consul/docs/k8s, b. Accessed: 2022-12-16.

Gossip in Consul. https://developer.hashicorp.com/consul/docs/architecture/gossip, a. Accessed: 2022-12-16.

Gossip Protocol. http://man.hubwiz.com/docset/Consul.docset/Contents/Resources/Documents/docs/internals/gossip.html, b. Accessed: 2022-12-16.

Kubernetes Basics. https://kubernetes.io/docs/tutorials/kubernetes-basics/, a. Accessed: 2022-12-16.

Multi-Cloud Kubernetes best practices. https://blog.scaleway.com/k8s-multi-cloud-best-practices/, b. Accessed: 2022-12-16.

Deploy Federated Multi-Cloud Kubernetes Clusters. https://developer.hashicorp.com/terraform/tutorials/kubernetes/multicloud-kubernetes. Accessed: 2022-12-16.

Terraform. https://developer.hashicorp.com/terraform, a. Accessed: 2022-12-16.

What is Terraform? https://developer.hashicorp.com/terraform/intro, b. Accessed: 2022-12-16.

Terraform Tutorial. https://www.youtube.com/watch?v=eE1LmKmuhm8, c. Accessed: 2022-12-16.

Understand Consul Service Mesh. https://developer.hashicorp.com/consul/tutorials/kubernetes-deploy/service-mesh. Accessed: 2022-12-16.

Flexera State of the Cloud Report 2022. https://resources.flexera.com/web/pdf/Flexera-State-of-the-Cloud-Report-2022.pdf, 2022. Accessed: 2022-12-16.

Parveen Bhandari. Multi-Cloud Kubernetes Solution and Strategy. https://www.xenonstack.com/use-cases/kubernetes-multi-cloud. Accessed: 2022-12-16.

Joseph Dhanapal. The Benefits and Limitations of a Multi-Cloud Strategy. https://www.pingidentity.com/en/resources/blog/post/multi-cloud-strategy-benefits-and-limitations.html, 2022. Accessed: 2022-12-16.

Morgan Perry. Kubernetes Multi-Cluster: Why and When To Use Them. https://www.qovery.com/blog/kubernetes-multi-cluster-why-and-when-to-use-them. Accessed: 2022-12-16.

Rags Srinivas and Janet Kuo. Virtual Panel: Kubernetes and the Challenges of Multi-Cloud. https://www.infoq.com/articles/Kubernetes-multi-cloud/. Accessed: 2022-12-16.

Chip Vaughn. Understanding and troubleshooting the different Gossip stages of Consul Nodes. https://support.hashicorp.com/hc/en-us/articles/9804284027283-Understanding-and-troubleshooting-the-different-Gossip-stages-of-Consul-Nodes. Accessed: 2022-12-16.