



Case Study: Integrating Azure IPv6 Private Link With Kubernetes

Meixing Le, Michael Wiederhold

Agenda



KubeCon



CloudNativeCon

North America 2020

Virtual

Brief intro to  **databricks** architecture

Inside the Unified Analytics Platform

Private link introduction

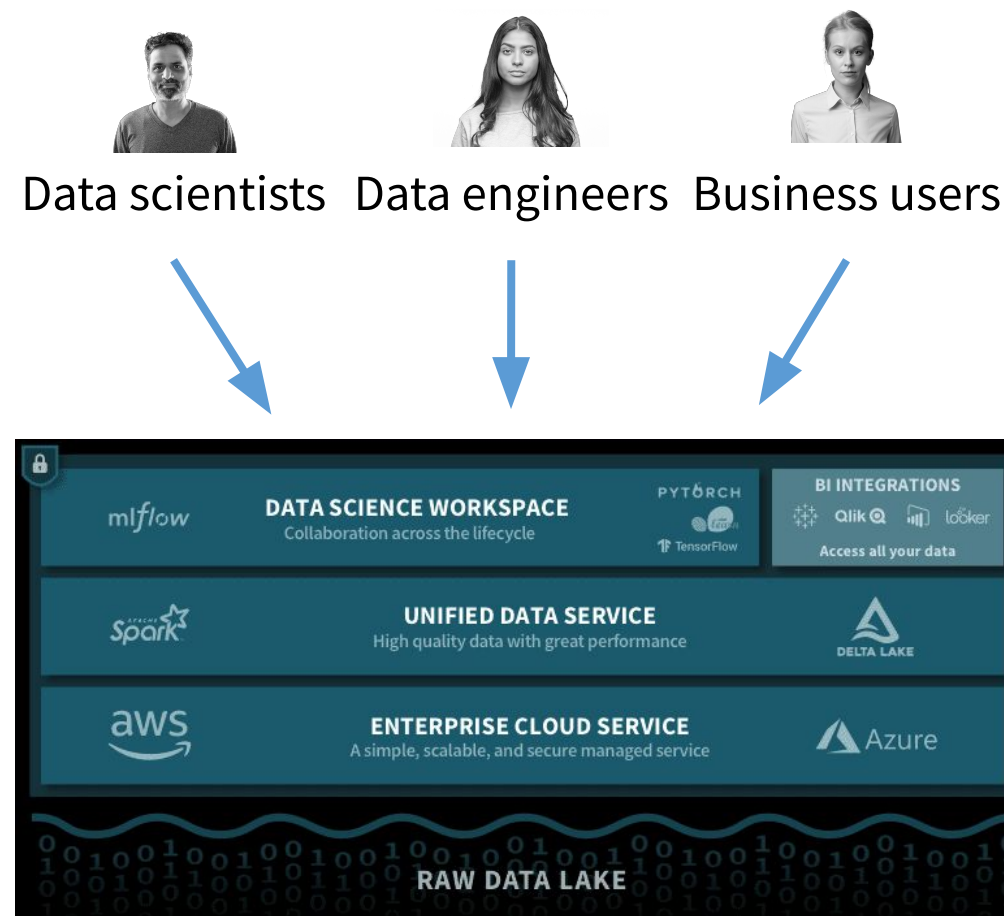
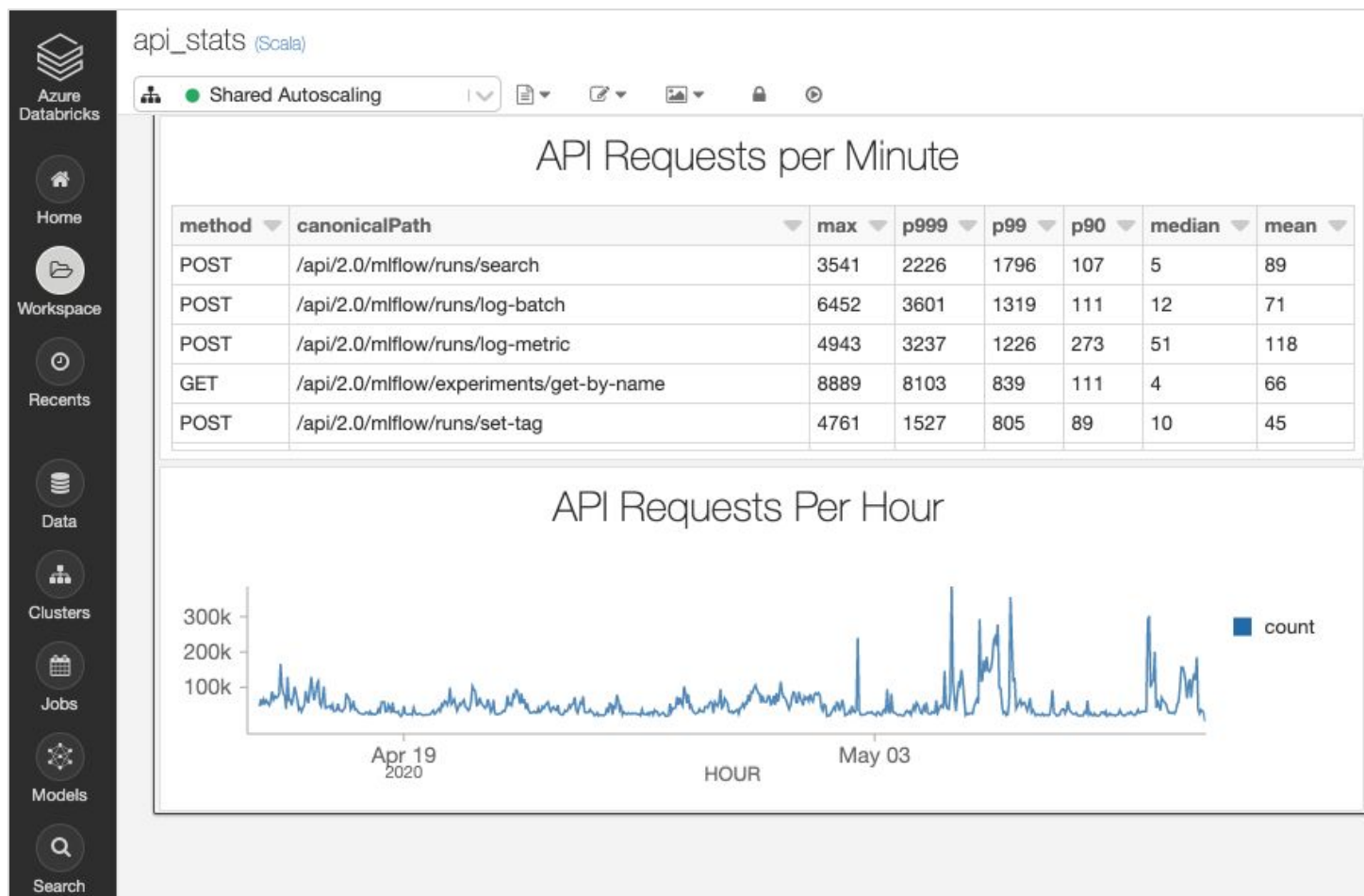
Journey to support Azure Private Link

Serving IPv6 traffic in Kubernetes without enabling IPv6 feature in Kubernetes

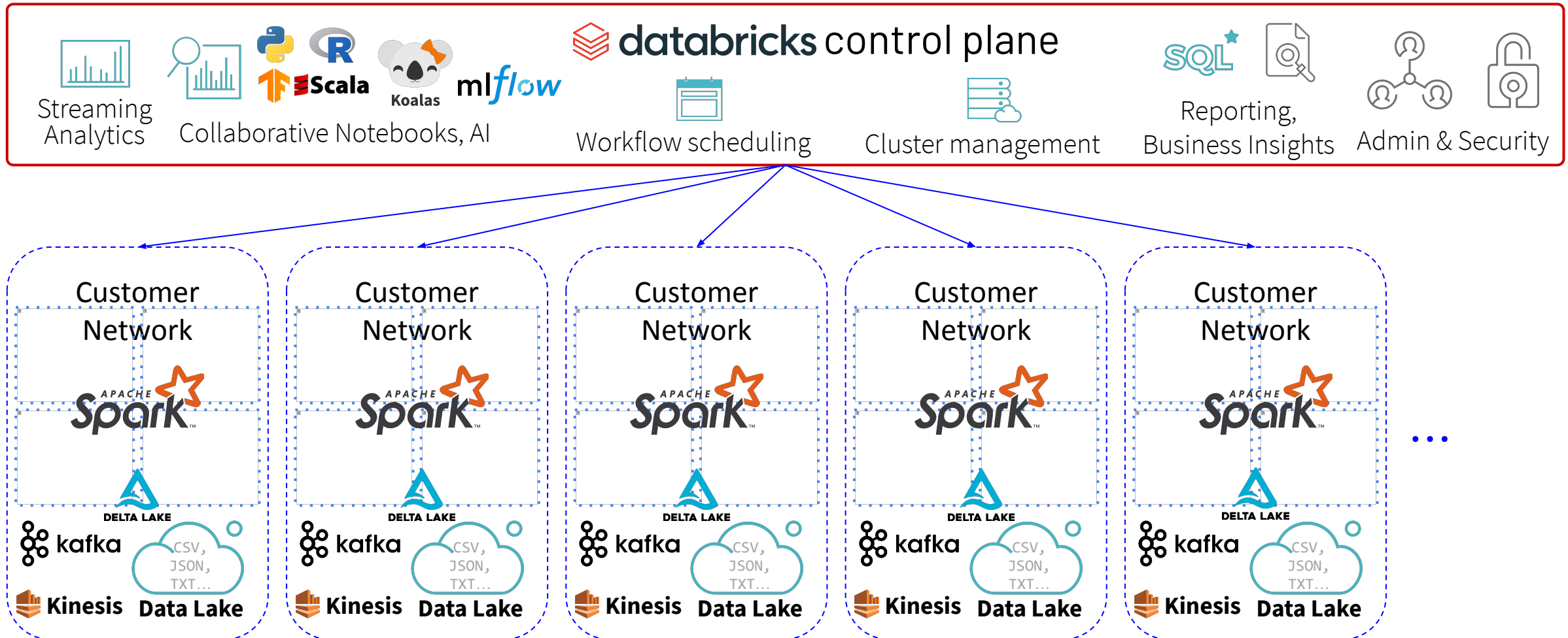
About databricks

- Founded in 2013 by the original creators of Apache Spark
- Data and AI platform as a service for 6000+ customers
- 1500+ employees, 300+ engineers, >\$350M annual recurring revenue

Product – unified analytics platform



Multiply by thousands of customers...



many regions on multiple clouds...



KubeCon



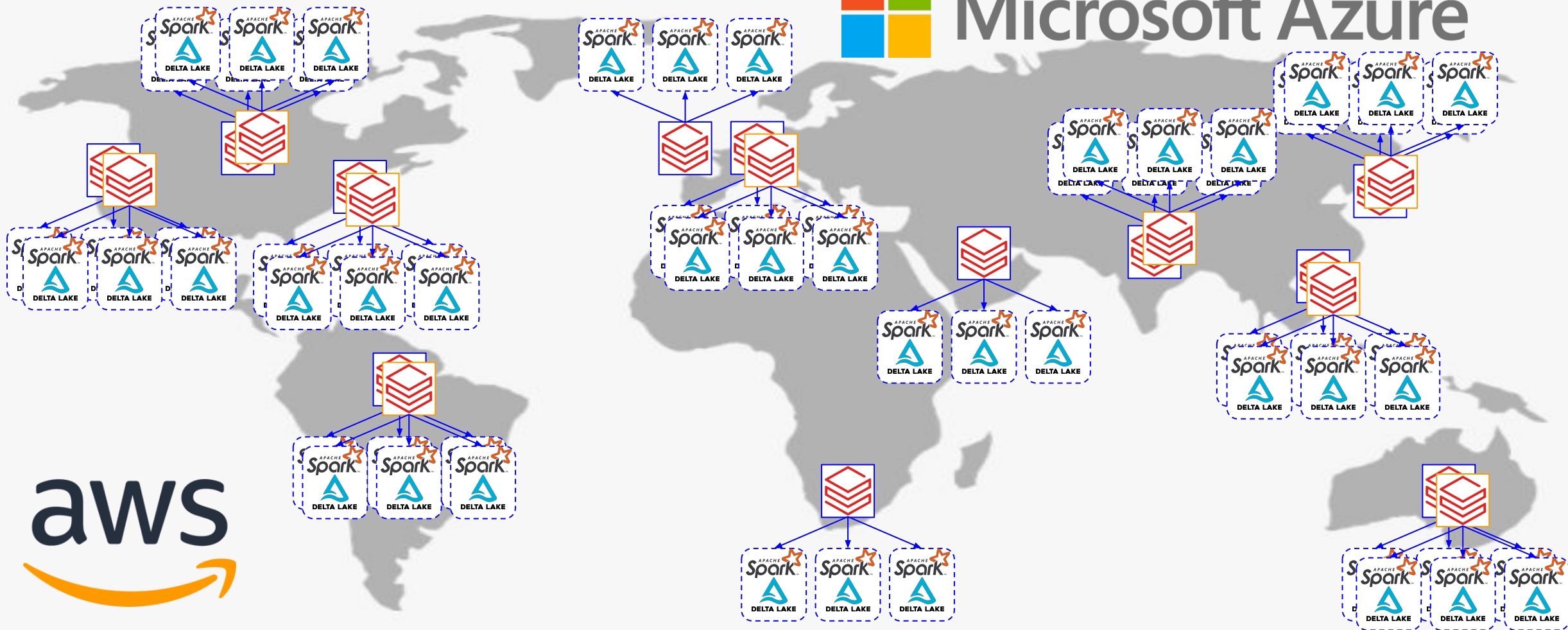
CloudNativeCon

North America 2020

Virtual



Microsoft Azure



That's the Databricks control plane



Virtual

North America 2020

100,000s of users

100,000s of Spark clusters per day

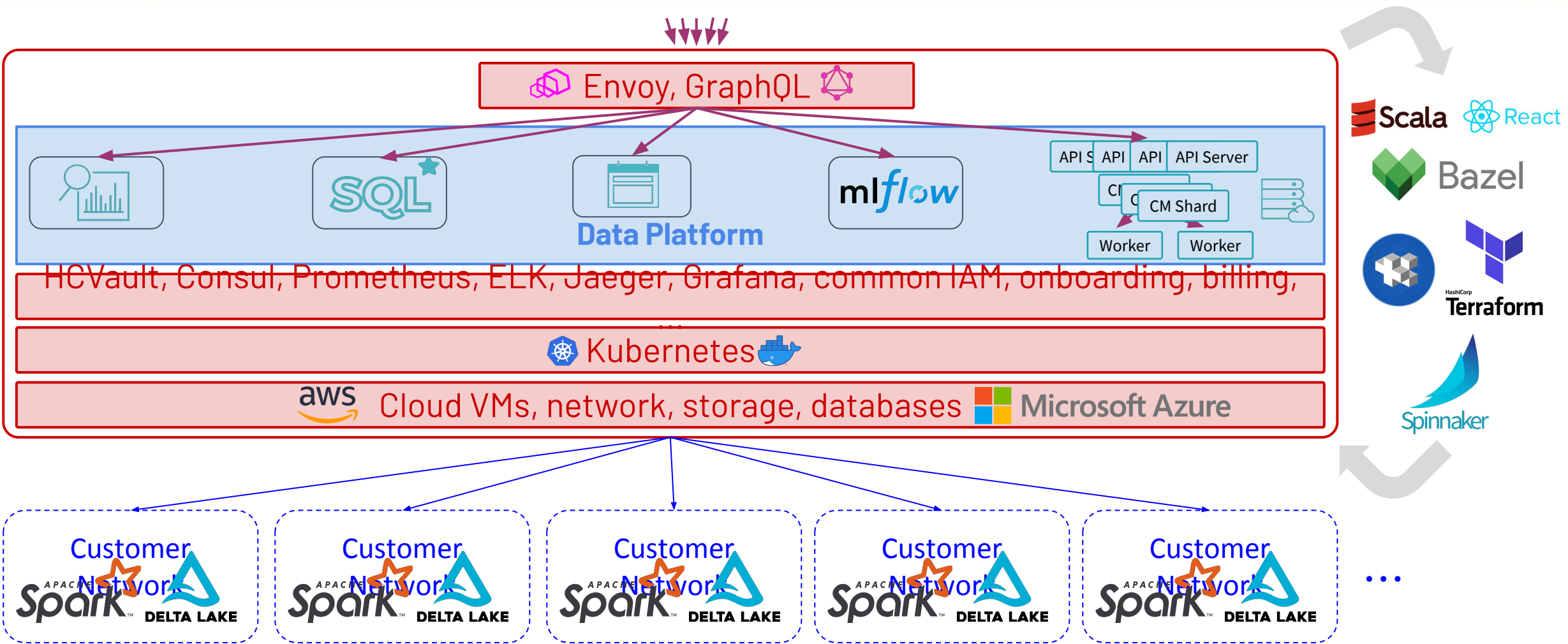
Millions of VMs launched per day

Exabytes of data processed per
day

3 deployment models

2000+ self managed
Kubernetes clusters

The Databricks data platform



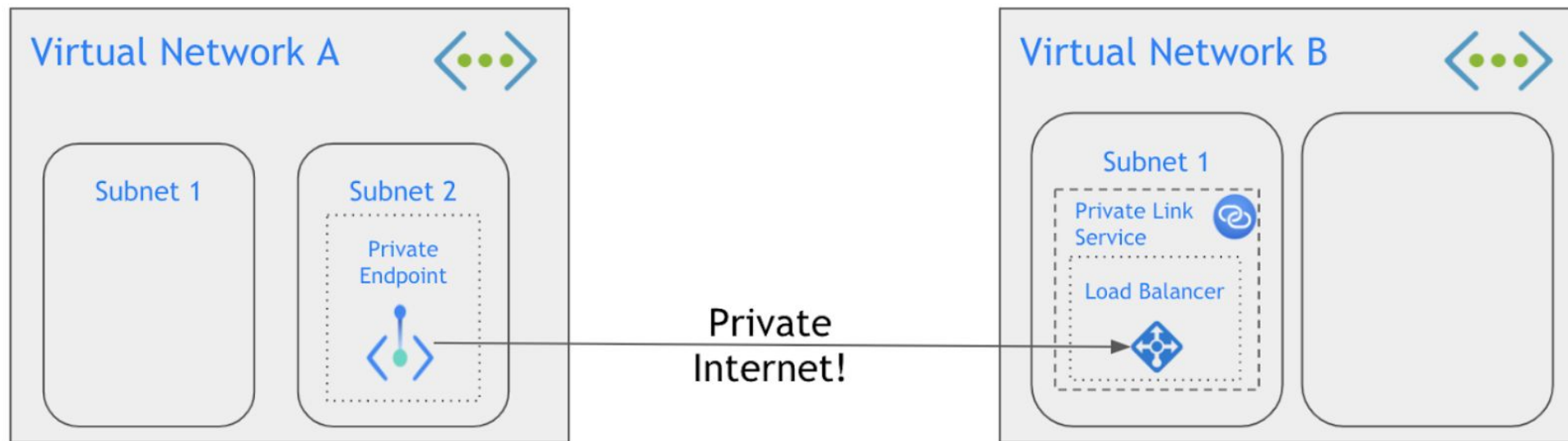
What is Private Link?

Traffic does not route via Internet to reach to services

Customers feel safe and private (Strong customer ask for SaaS products)

All major cloud providers offer this feature

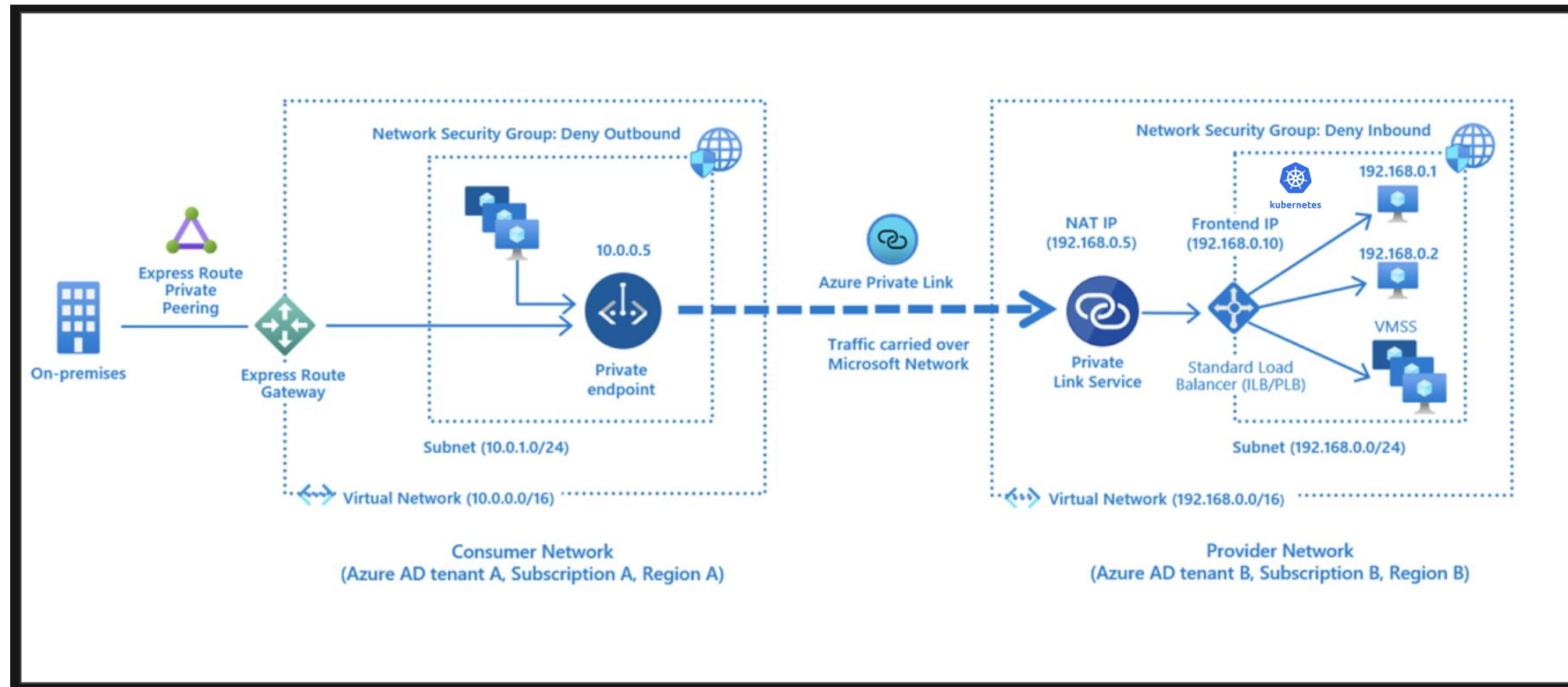
Azure Private Link allows secure communication between private networks over Azure's private internet



Private Endpoint

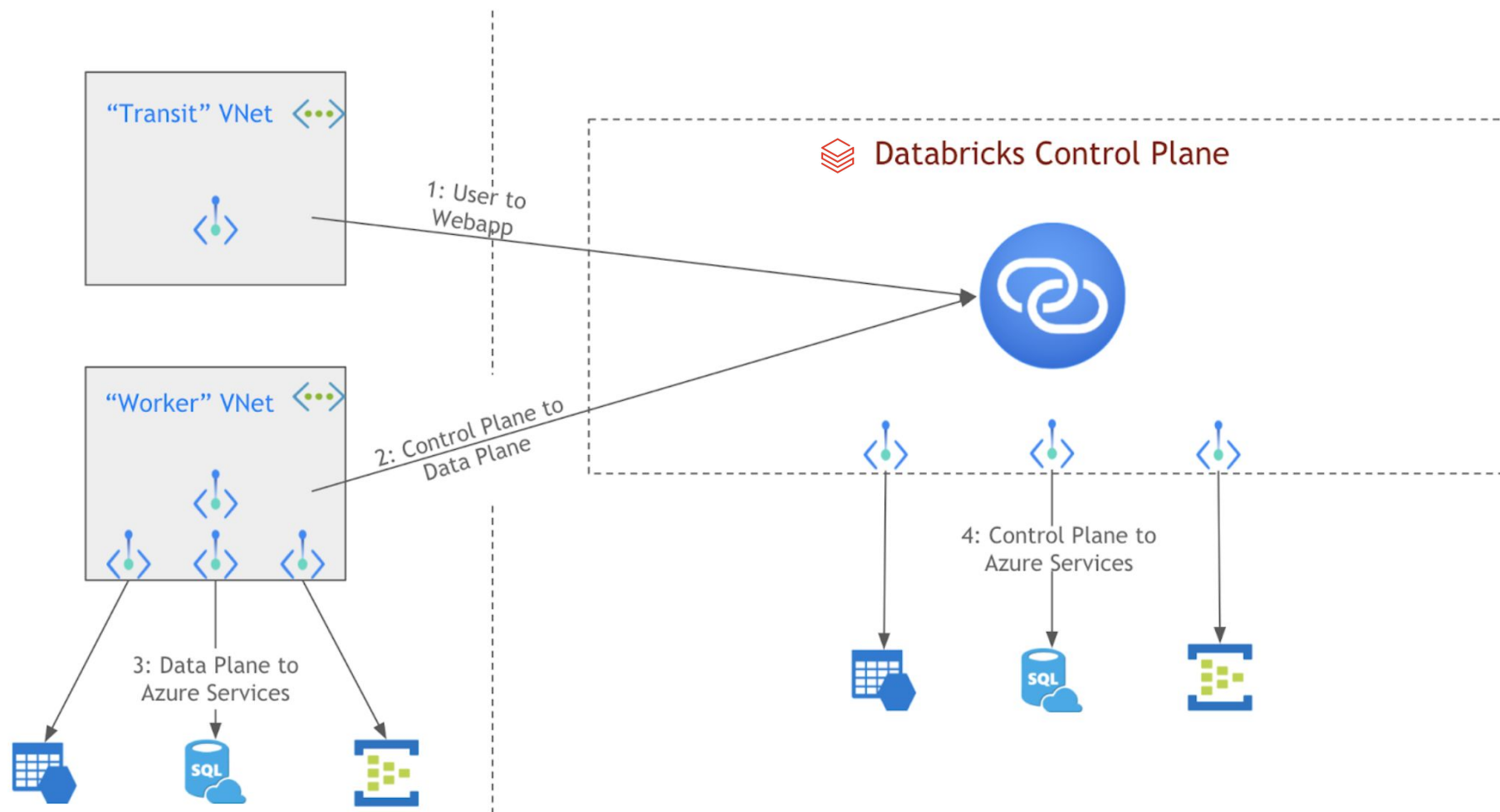
- A network interface that connects privately a Private Link service
- Access to service behind Load balancer via Cloud provider networks

Azure Private Link



<https://docs.microsoft.com/en-us/azure/private-link/private-link-service-overview>

databricks use cases




Private Link Infra: IPv6 support



Virtual

North America 2020

Databricks is a first party service on Azure

-  [Azure Databricks](#)
 - Databricks appears as a native service in Azure
 - Creating a Databricks workspace is as easy as creating VMs, databases in Azure
- Azure provides two private link support models
 - Third party offering – available to all Azure customers
 - PaaS version of Private Link (deeper integration)
 - It appears to customers as routed over IPv4
 - Traffic is carried over IPv6 between VNets
- Azure IPv6 support: VNet, subnet, LB, VMSS (virtual machine scale set) all support dual-stack
- Requirement: Control plane needs to accept IPv6 traffic

Challenge:

- Control plane services are completely running on Kubernetes
- 2 high level options
 - Proxy: convert IPv6 to IPv4 outside Kubernetes
 - Enable IPv6 on Kubernetes

Kubernetes at databricks



KubeCon




CloudNativeCon



North America 2020

Virtual

Databricks control plane is not using managed Kubernetes (not AKS)

- Databricks is multi-cloud, we want to be consistent across different cloud providers
 - Always same OS, kernel, Kubernetes versions on all cloud providers
 - More controls over the whole infrastructure, easier to support our own services
- Bake Unified Kubernetes VM Image (UKI) and VMs can self bootstrap into Kubernetes clusters
 - Bootstrap secrets rely on HC-vault 

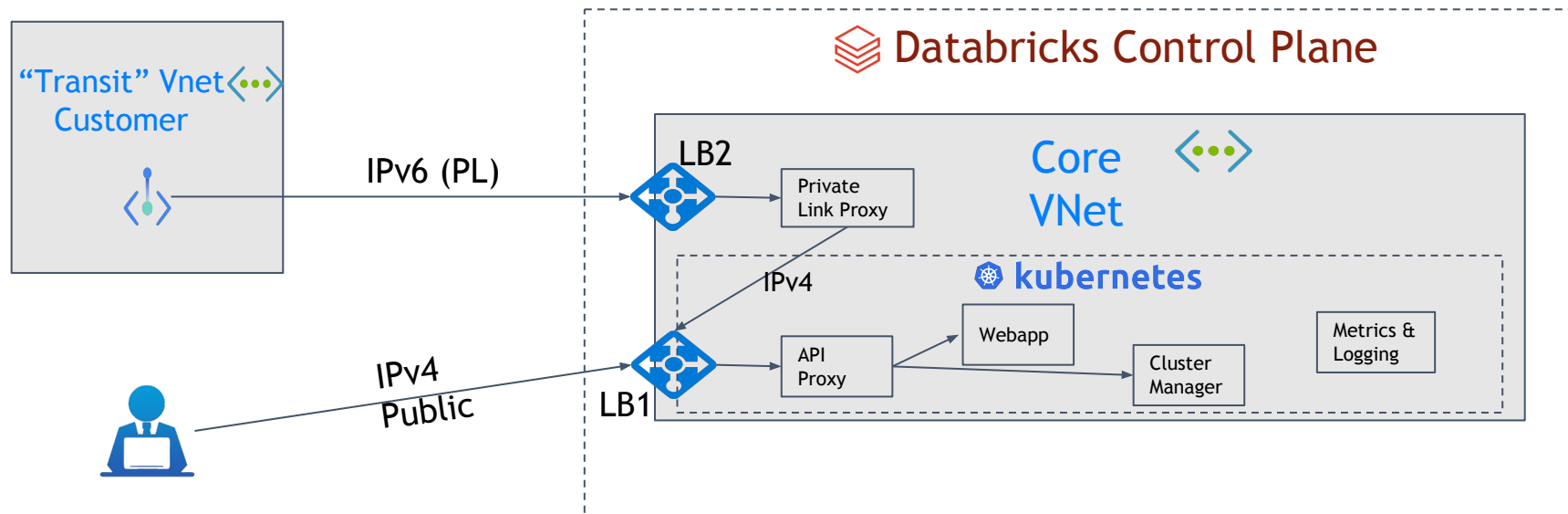
Configurations

- Disable IPv6 in the kernel
- CNI:  flannel
- Container runtime:  docker
- Kubernetes version: V1.16
- Load balancer in Azure: One single LB, each k8s service adds LB rules

Option 1: Proxy outside Kubernetes

Proxy solution

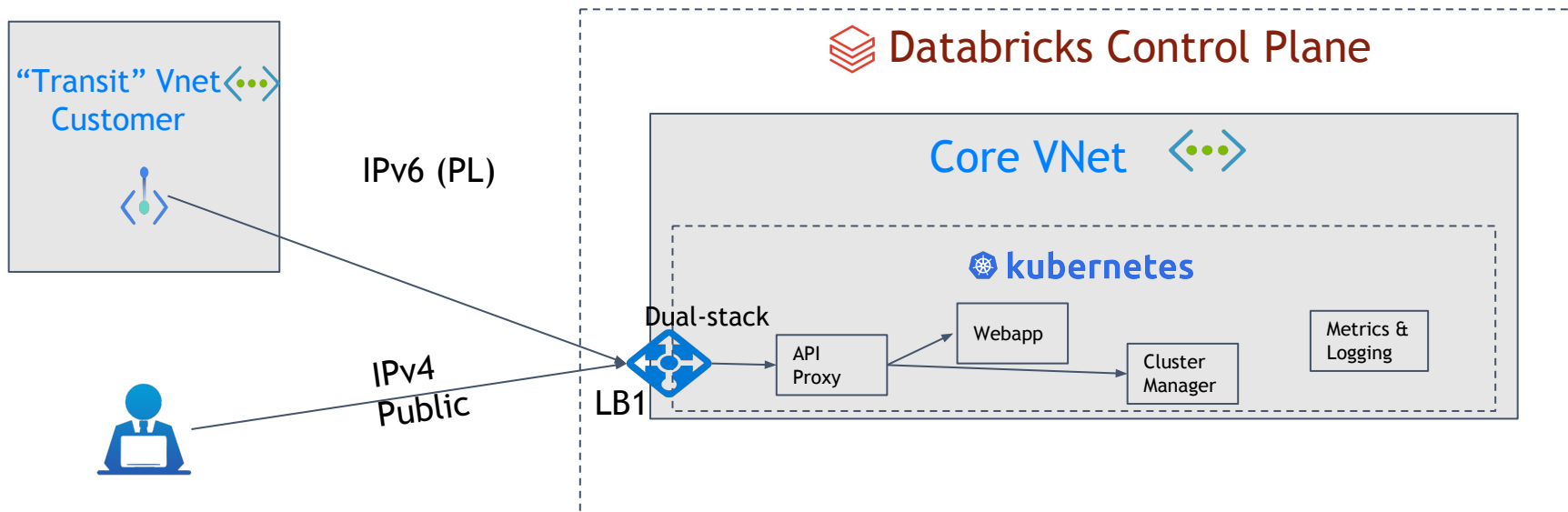
- Used by some Azure internal services
 - A dedicated Load balancer (LB2) / VMSS outside Kubernetes as Kubernetes cannot accept IPv6
 - Proxy terminates IPv6 and use IPv4 to talk to Kubernetes LB (LB1)
- Challenges:
 - How to deploy? VM image or container?
 - VM: one more image to manage, Container: docker commands via scripts?
 - How to monitor? Metrics & logging?



Option 2: Kubernetes support IPv6

Native IPv6 support

- Kubernetes needs to enable dual-stack feature
- Simpler architecture
- Challenges:
 - Stability: Alpha feature in k8s v1.16 (beta targeting v1.20)
 - Overkill: Only several frontend pods/services needs IPv6 support
 - Could be huge engineering effort (prototyping, testing, etc)



Dual-stack on Kubernetes



North America 2020

- Not to be confused with IPv6 single stack feature which entered alpha in v1.9 and moved to beta in v1.18
- Alpha feature starting v1.16 but mostly stable
- Assign both IPv4 and IPv6 to every pods, but each Service must be for a single address family
- Not entering beta due to pending discussions on service APIs
- Networking prerequisites
 - Kubernetes nodes / host level networking must support dual stack (Azure VMSS does support)
 - CNI must support dual stack (flannel doesn't support)
 - Kubenet and Calico should support dual stack (vary by cloud provider)
- Proxy solution seems a better option for our use case in the short term

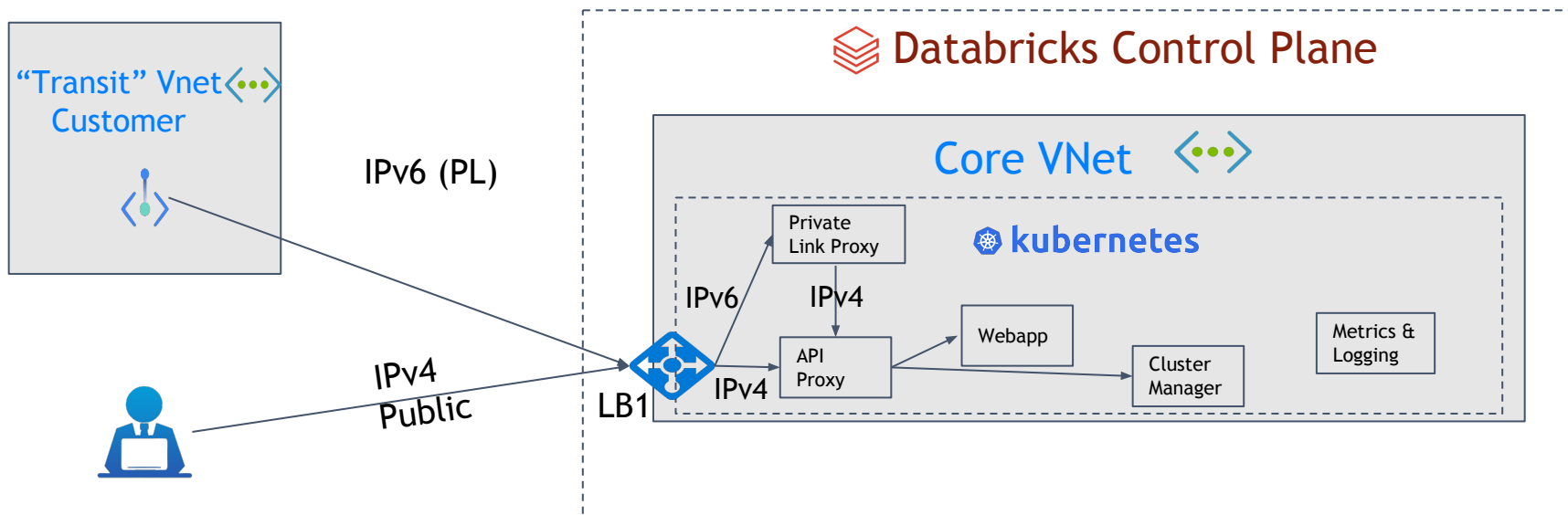
Option3: Combining the 2 options?

Revisit proxy solution, can we move the proxy into Kubernetes?

- We can get deployment and monitoring for free
- LB does support dual-stack, 2 LBs can be combined into one
- VMSS also support dual-stack, only Kubernetes (flannel) does not support

How about deploy the proxy as pods using Kubernetes and give VM networking to the pod?

- Will host networking work for dual-stack? It should?



Solution



- Provision IPv6 CIDR on VNet / Subnet
- Provision private link IPv6 IP and attach to the LB used by Kubernetes
- Provision a VMSS with Kubernetes VM image that enables IPv6
 - Add IPv6 to VMSS nic to make it dual-stack
 - If using Terraform
 - Azure provider 2.0
 - Use Terraform resource `azurerm_linux_virtual_machine_scale_set`
 - Else
 - Use AZ CLI to add IPv6 after VMSS provisioned (before scale up instances)
 - This VMSS bootstraps into a dedicated node pool in Kubernetes
 - Add the VMSS to a separate LB backend pool
 - All the IPv6 IPs on VMs are internal IP, public IPv6 IP can only put on the LB
- Deploy privatelink-proxy as a general Kubernetes deployment to the dedicated node pool
 - Setup proxy pods to use host networking, it will use eth0 of the VM and get both IPv4 and IPv6
 - nginx proxy listens on port 443
 - If PodSecurityPolicy is enabled, **HostNetwork** needs to be configured in the policy.

Solution (cont'd)

- Configure LB rules to load balance traffic to proxy pod
 - This is usually handled by Kubernetes, but we set it up this time
- Make sure to whitelist private link IPv6 traffic in the Network Security Groups
- Proxy works as a charm!

Benefits:

- Straightforward proxy solution, easy to troubleshoot
- Proxy deployment managed by Kubernetes, easy to update
- Proxy deployment easy to scale by Kubernetes as traffic load increase
- Existing monitoring and logging infrastructure can automatically cover the proxy
- Same solution for Data plane to Control plane traffic

blue-minion-00001r	Ready
blue-minion-00001s	Ready
blue-privatelink-minion-000000	Ready
blue-privatelink-minion-000001	Ready
blue-privatelink-minion-000003	Ready
green-elk-minion-000000	Ready
green-elk-minion-000001	Ready

Dedicated node pool “privatelink” in Kubernetes
A VMSS accepts both IPv4 and IPv6 traffic

```
name: privatelink-nginx-metrics-exporter
ports:
- containerPort: 7779
  hostPort: 7779
  name: info-service
  protocol: TCP
resources:
  limits:
    cpu: 100m
    memory: 500Mi
  terminationMessagePath: /dev/termination-log
  terminationMessagePolicy: File
  dnsPolicy: ClusterFirstWithHostNet
  hostNetwork: true
  nodeSelector:
    pool: privatelink
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: NoSchedule
    key: pool
    operator: Equal
    value: privatelink
```

Proxy template:

hostNetwork: true

Scheduled to “privatelink” node pool

Dual-stack pod / VM



KubeCon



CloudNativeCon

North America 2020

Virtual

nginx pod uses hostNetwork Dual-stack

```
[centos@master ~]$ kubectl exec -it privatelink-proxy-6fffc67758-ddln7 bash -n production
Defaulting container name to privatelink-nginx.
Use 'kubectl describe pod/privatelink-proxy-6fffc67758-ddln7 -n production' to see all of the
root@blue-privatelink-minion-000001:/databricks# ifconfig
cni0
Link encap:Ethernet HWaddr 3e:ed:8c:fa:51:69
inet addr:10.2.49.1 Bcast:10.2.49.255 Mask:255.255.255.0
inet6 addr: fe80::3ced:8cff:fefa:5169/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1472 Metric:1
RX packets:31058613 errors:0 dropped:0 overruns:0 frame:0
TX packets:42736387 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:182658630077 (182.6 GB) TX bytes:5591124457 (5.5 GB)

eth0
Link encap:Ethernet HWaddr 00:0d:3a:77:cd:f1
inet addr:10.120.8.9 Bcast:10.120.11.255 Mask:255.255.252.0
inet6 addr: fe80::20d:3aff:fe77:cdf1/64 Scope:Link
inet6 addr: ace:cab:deca:deed::9/128 Scope:Global
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:193113002 errors:0 dropped:0 overruns:0 frame:0
TX packets:175230449 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:54606169589 (54.6 GB) TX bytes:203253720369 (203.2 GB)
```

```
Last login: Wed Sep 16 08:01:08 2020 from master.internal.cloudapp.net
[centos@blue-privatelink-minion-000001 ~]$ ifconfig
cni0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1472
inet 10.2.49.1 netmask 255.255.255.0 broadcast 10.2.49.255
inet6 fe80::3ced:8cff:fefa:5169 prefixlen 64 scopeid 0x20<link>
ether 3e:ed:8c:fa:51:69 txqueuelen 1000 (Ethernet)
RX packets 31055693 bytes 182643444049 (170.0 GiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 42732241 bytes 5590497997 (5.2 GiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.120.8.9 netmask 255.255.252.0 broadcast 10.120.11.255
inet6 fe80::20d:3aff:fe77:cdf1 prefixlen 64 scopeid 0x20<link>
inet6 ace:cab:deca:deed::9 prefixlen 128 scopeid 0x0<global>
ether 00:0d:3a:77:cd:f1 txqueuelen 1000 (Ethernet)
RX packets 193094268 bytes 54600781181 (50.8 GiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 175213569 bytes 203236706812 (189.2 GiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Node level host networking has the same dual-stack interfaces

IPv6 on Azure LB



Virtual

North America 2020

kubernetes-standard | Frontend IP configuration

Search (Cmd+ /)	<<	+ Add	Refresh
Overview	a13a0a5435f7a4a1c8d5ca5b166d2d4e	20.194.14.147 (kubernetes-a13a0a5435f7a4a1c8d5ca5b166d2d4e)	1
Activity log	aee6f5d7ad10c4065a2d37d29df32a03	20.194.15.187 (kubernetes-aee6f5d7ad10c4065a2d37d29df32a03)	1
Access control (IAM)	ae4e048380e55491b84ef26b9b5d45e9	20.194.14.253 (kubernetes-ae4e048380e55491b84ef26b9b5d45e9)	1
Tags	a0b2efd15ec5e435f8c8f44c6f691076	20.194.10.70 (kubernetes-a0b2efd15ec5e435f8c8f44c6f691076)	3
Diagnose and solve problems	a8a03f251e6474cd488e886df6108961	20.194.15.48 (kubernetes-a8a03f251e6474cd488e886df6108961)	3
Settings	a19bca00ff44d4dbaa8792fd28e9f6c2	20.194.15.88 (kubernetes-a19bca00ff44d4dbaa8792fd28e9f6c2)	3
Frontend IP configuration	a53ca298539a0446ea23a6d261d71b5b	20.194.15.68 (kubernetes-a53ca298539a0446ea23a6d261d71b5b)	3
Backend pools	aa9b1a313c79649368441d33b7807f5f	20.194.12.150 (kubernetes-aa9b1a313c79649368441d33b7807f5f)	1
Health probes	ae289823c427b43b0b46fb49c0d804f3	20.194.15.118 (kubernetes-ae289823c427b43b0b46fb49c0d804f3)	1
Load balancing rules	a168f8cd55dd24ba796e682fd2472cf2	20.194.15.174 (kubernetes-a168f8cd55dd24ba796e682fd2472cf2)	2
Inbound NAT rules	a7eef487e98124dcaae9d8f92e6b40d	20.194.14.154 (kubernetes-a7eef487e98124dcaae9d8f92e6b40d)	2
Outbound rules	a956ed1b7509f4304b081733877d7e3e	20.194.12.218 (kubernetes-a956ed1b7509f4304b081733877d7e3e)	1
Properties	abc16729efab44280a883d6fb5dc2e2b	20.194.11.118 (kubernetes-abc16729efab44280a883d6fb5dc2e2b)	1
Locks	a09c54ce5b2bb4163b8538dfc951059e	20.194.15.22 (kubernetes-a09c54ce5b2bb4163b8538dfc951059e)	2
Monitoring	abac5b01a5fa04f209d90db92b1fa1e9	20.194.12.10 (kubernetes-abac5b01a5fa04f209d90db92b1fa1e9)	1
Alerts	a662283df12824b889d58ca959273fda	20.194.11.36 (kubernetes-a662283df12824b889d58ca959273fda)	3
Metrics	aaa777e1542e44294a06d79c9db9df94	20.196.64.169 (kubernetes-aaa777e1542e44294a06d79c9db9df94)	1
Insights (preview)	a53957d6c48f5415bb5e13923ca6d6c7	40.82.153.191 (kubernetes-a53957d6c48f5415bb5e13923ca6d6c7)	1
Automation	aca2c9b0b771f4c93a7851878db3fdbf	20.41.104.28 (kubernetes-aca2c9b0b771f4c93a7851878db3fdbf)	1
Tasks	aad19681a92ea4ce8ba53002e7066f3c	20.41.104.207 (kubernetes-aad19681a92ea4ce8ba53002e7066f3c)	1
Export template	adfc77570ed514c03834d8d9209029c9	20.41.104.149 (kubernetes-adfc77570ed514c03834d8d9209029c9)	1
	privatelink_ipv6	2603:10e1:100:2::1427:bdd2 (privatelink2_ipv6)	2

Azure Databricks via Private link



KubeCon



CloudNativeCon

North America 2020

Virtual

Administrator: Command Prompt

Microsoft Windows [Version 10.0.17763.1282]
(c) 2018 Microsoft Corporation. All rights reserved.

```
C:\Users\jake>nslookup adb-145393161930121.1.staging.azuredatabricks.net
Server: UnKnown
Address: 168.63.129.16
```

Non-authoritative answer:

```
Name: adb-145393161930121.1.privatelink.staging.azuredatabricks.net
Address: 10.0.0.6
Aliases: adb-145393161930121.1.staging.azuredatabricks.net
```

```
C:\Users\jake>nslookup adb-145393161930121.1.staging.azuredatabricks.net
Server: UnKnown
Address: 168.63.129.16
```

Non-authoritative answer:

```
Name: adb-145393161930121.1.privatelink.staging.azuredatabricks.net
Address: 10.0.0.4
Aliases: adb-145393161930121.1.staging.azuredatabricks.net
```

C:\Users\jake>

+ Clear-ClientDnsCache

+ ~~~~~

```
+ CategoryInfo          : ObjectNotFound: (Clear-ClientDnsCache)
+ FullyQualifiedErrorId : CommandNotFoundException
```

```
PS C:\Users\jake> Clear-DnsClientCache
PS C:\Users\jake> Clear-DnsClientCache
PS C:\Users\jake> Clear-DnsClientCache
PS C:\Users\jake>
```

Recap

- If you need IPv6 traffic support in IPv4 Kubernetes
 - Enable IPv6 (Dual-stack) everywhere on Cloud provider Infra
 - Vnet, Subnet, LB, VMSS
 - IPv6 to IPv4 proxy
 - Deploy as regular Kubernetes deployment
 - Use hostNetwork to receive IPv6 traffic and proxy
- AzureDatabricks private link is in private preview
 - Also available in Azure GovCloud!

Thank you

