Microsoft

Cross-Cloud Connectivity

Diego Casati – Senior Software Engineer
Commercial Software Engineering

```
DCASATI(1)                  BSD General Commands Manual                  DCASATI(1)

NAME
     dcasati -- Diego Casati: a (curious) Engineer

SYNOPSIS
     dcasati [--network-engineering] [--software-define-(network|storage)]
             [--software-development] [--orchestration] [--devops]

HISTORY

     Diego Casati is a Senior Software Engineer for Microsoft focusing on
     Kubernetes, Networking and Linux and BSDs. Prior to his current role
     at Microsoft, he spent over a decade working in the Telco and IT
     industries at various capacities, from Networking Engineering to Systems
     Engineering and Security Specialist.

     He is a strong proponent of free and open source solutions, advocating
     for the use of BSDs to connect all things.

     When not hacking on computers you can catch him spending time playing with
     his 1-year-old son.

SEE ALSO
    - dcasati @github
    - diegocasati @Twitter

BUGS
   Too many to list here ;)
```


CROSS_CLOUD
THINGS

Microsoft

Cross-Cloud Connectivity

What worked, what failed and lessons learned.

Key objective

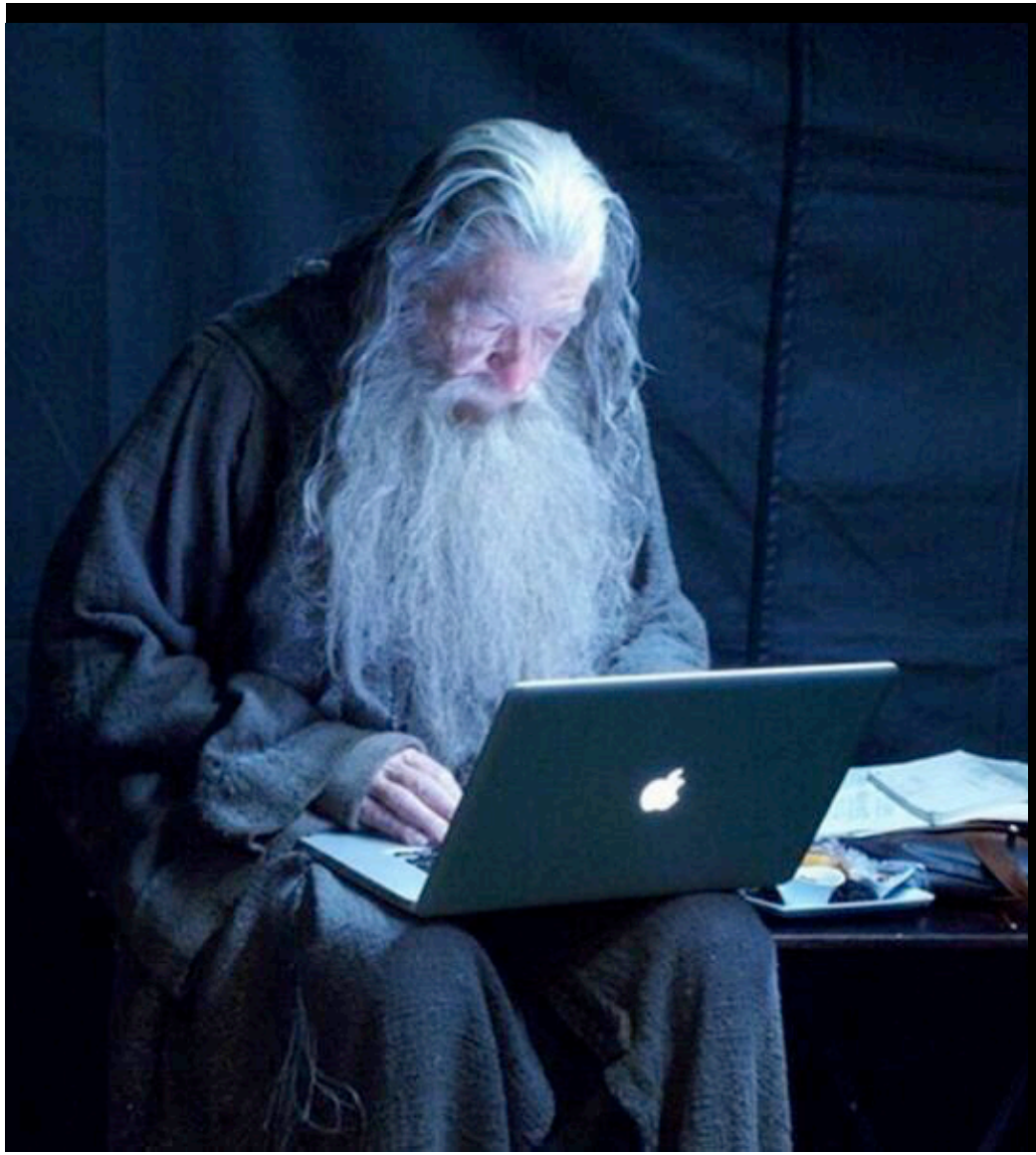
Ignite your curiosity by showing a real use case of an engineering exercise.

Main()

What?
Why?
Where?
How?

What?

Deploy Cassandra on top of Kubernetes *spanning two clouds*



Wait? What?
Let me check
online

Why?

Proactive exercise simulating Customers looking to (1) migrate or have their workload (2) spanning in two clouds

Why?

(1) **migrate** or have their workload

- For replication (e.g.: backup)
- To comply with a scheduled downtime (e.g.: keep a service running by moving data between Data Centers)

Why?

(2) **spanning** in two clouds

- To increased availability
- To provide load balancing
- To connect to an existing Data Center (e.g.: on premise)

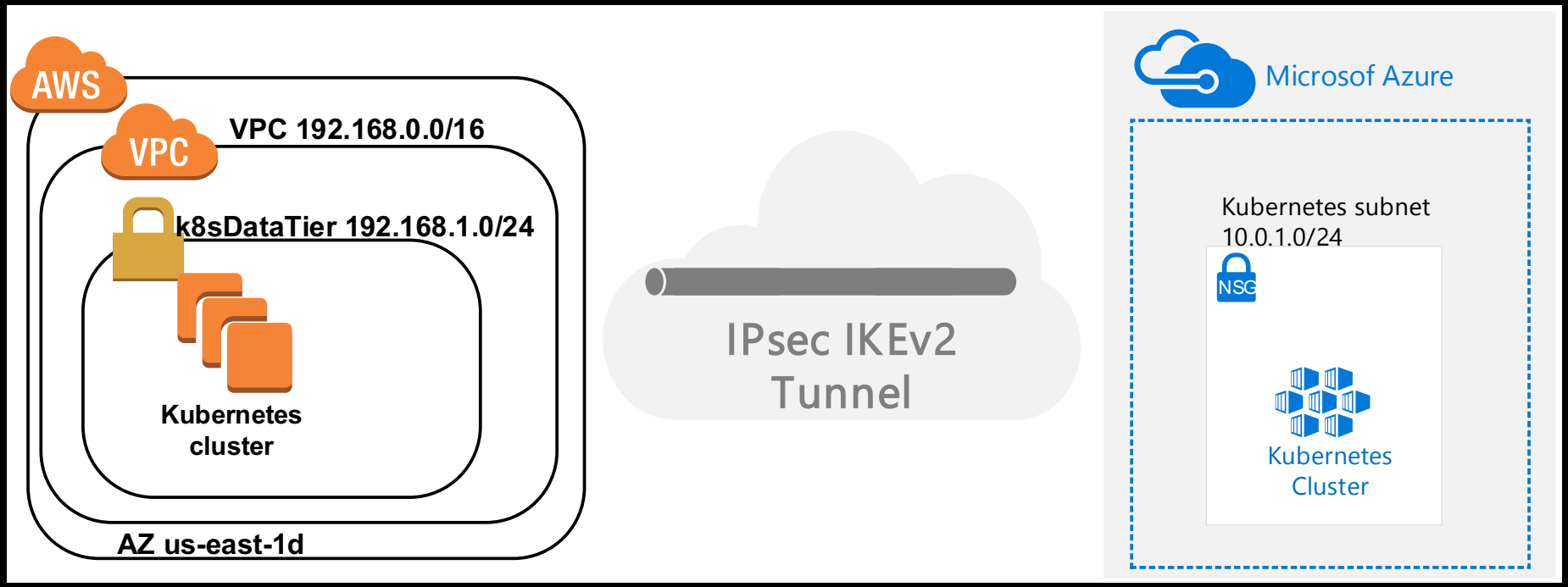
Where?

AWS
Azure

How?

site-to-site VPN between the clouds using open source components

END-TO-END TOPOLOGY - NETWORK

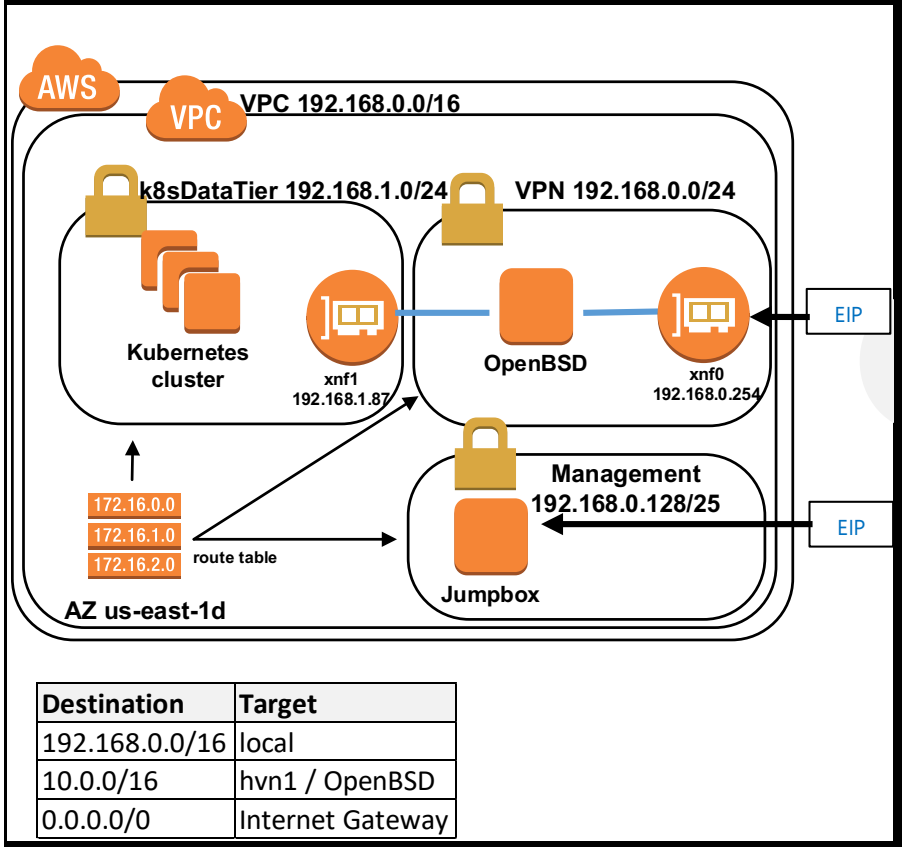



Dude...wait....
What???
ICANHASCHEEZBURGER.COM

Common Vocabulary

Acronym	Explanation
Jumpbox	A jump server is a hardened and monitored device that spans two dissimilar security zones and provides a controlled means of access between them.
VPN	A **virtual private network (VPN)** extends a private network across a public network, and enables users to send and receive data across shared or public networks as if their computing devices were directly connected to the private network.

AWS Vocabulary

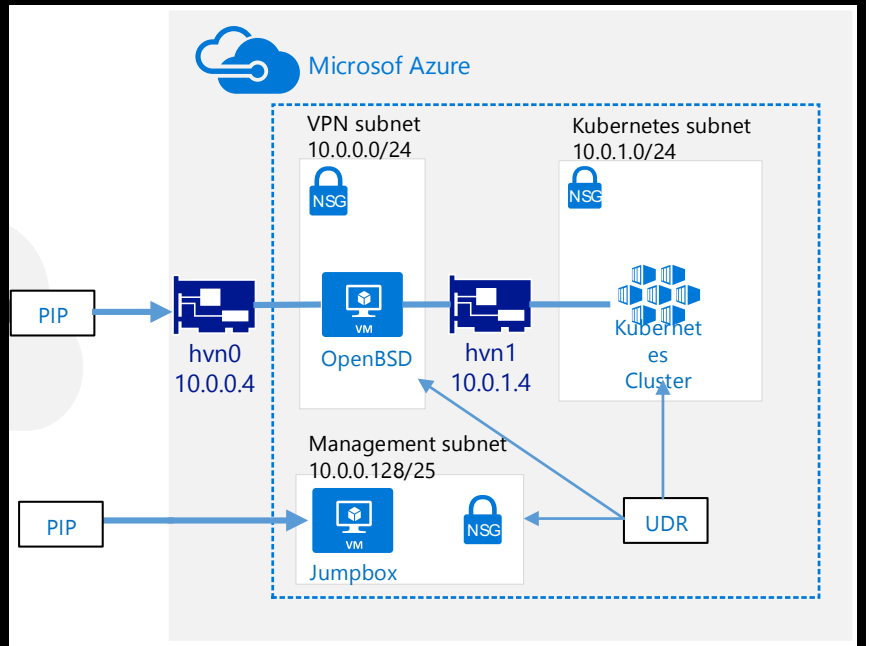


Destination	Target
192.168.0.0/16	local
10.0.0.0/16	hvn1 / OpenBSD
0.0.0.0/0	Internet Gateway

Acronym	Explanation
VPC	A *virtual private cloud* (VPC) is a virtual network dedicated to your AWS account.
EIP	An *Elastic IP address* is a static, public IPv4 address designed for dynamic cloud computing.
Security Group	A *security group* acts as a virtual firewall for your instance to control inbound and outbound traffic

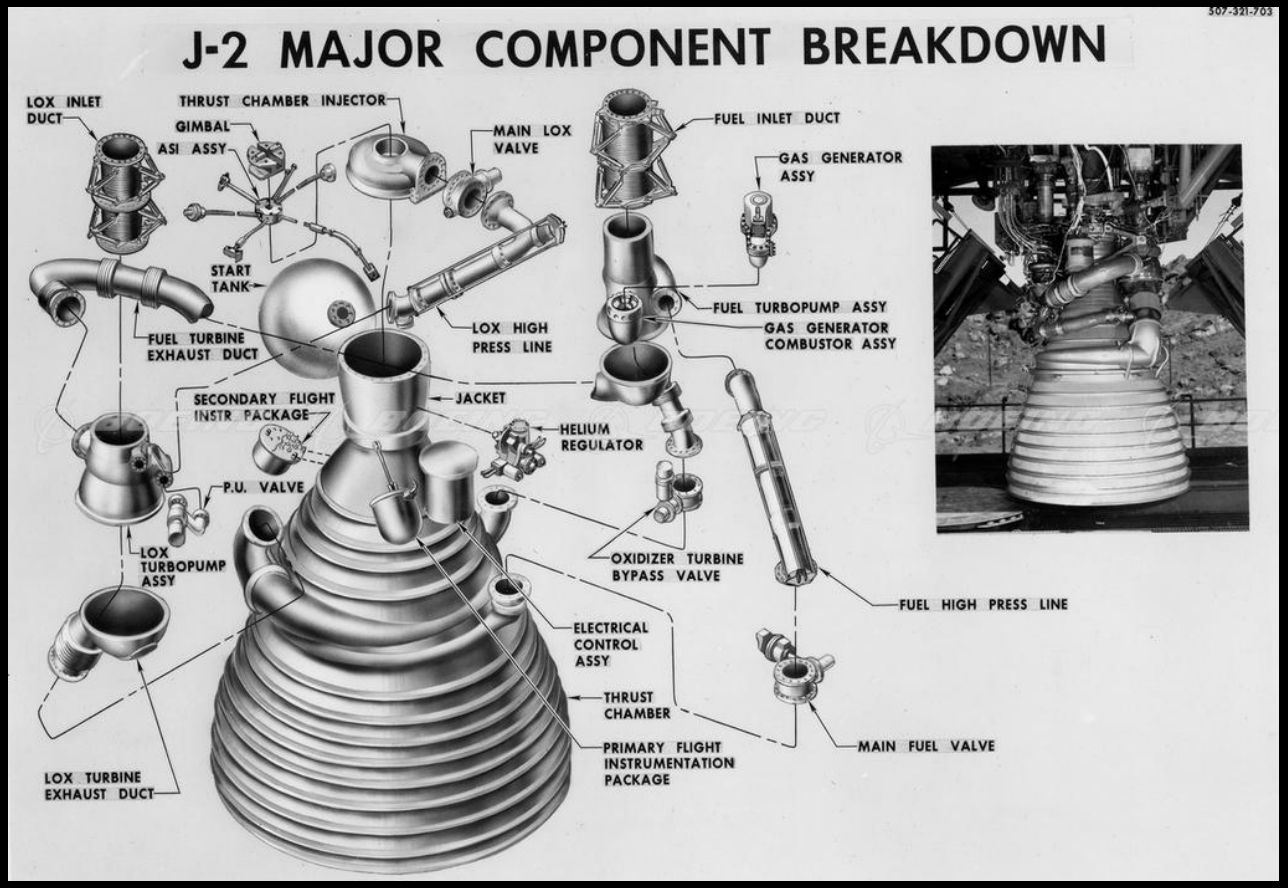
Azure Vocabulary

Acronym	Explanation
VNET	Azure Virtual Network enables Azure resources to communicate with each other and the internet.
PIP	A PIP is a *public instance-level IP address* associated with the VM in addition to the VIP.
NSG	A network security group (NSG) contains a list of security rules that allow or deny network traffic to resources connected to Azure Virtual Networks (VNet).
UDR	User defined routes



Destination	Target	Notes
192.168.0.0/16	10.0.1.4	OpenBSD's internal NIC

Let’s breakdown the solution



The solution – in pieces

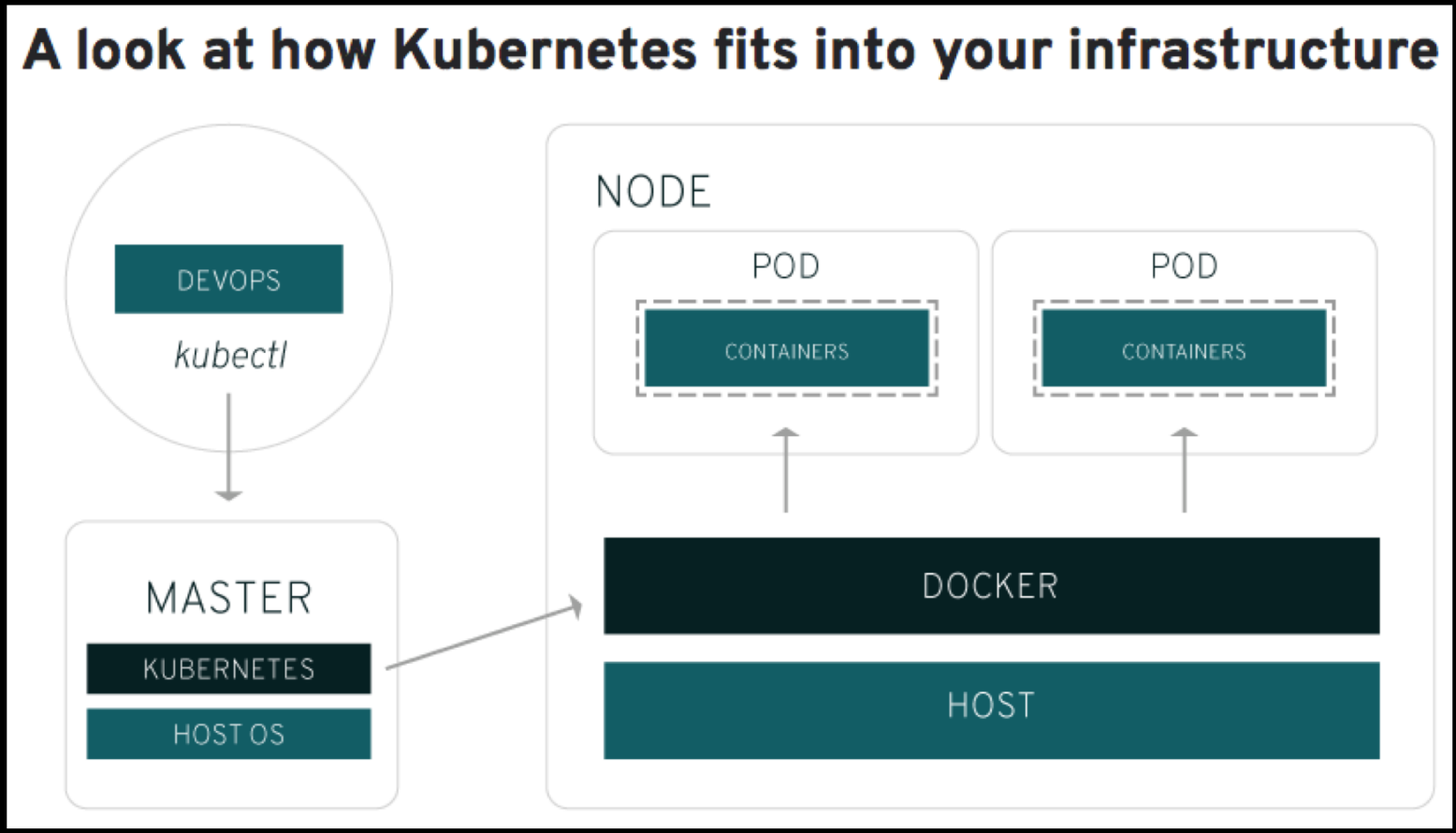
Solution	AWS	Azure
Cassandra	Kubernetes YAML files	Helm Chart
Kubernetes	KOPS	ACS-Engine
Network	VPN: OpenVPN? StrongSwan? Other IPSec solution?	

Cassandra

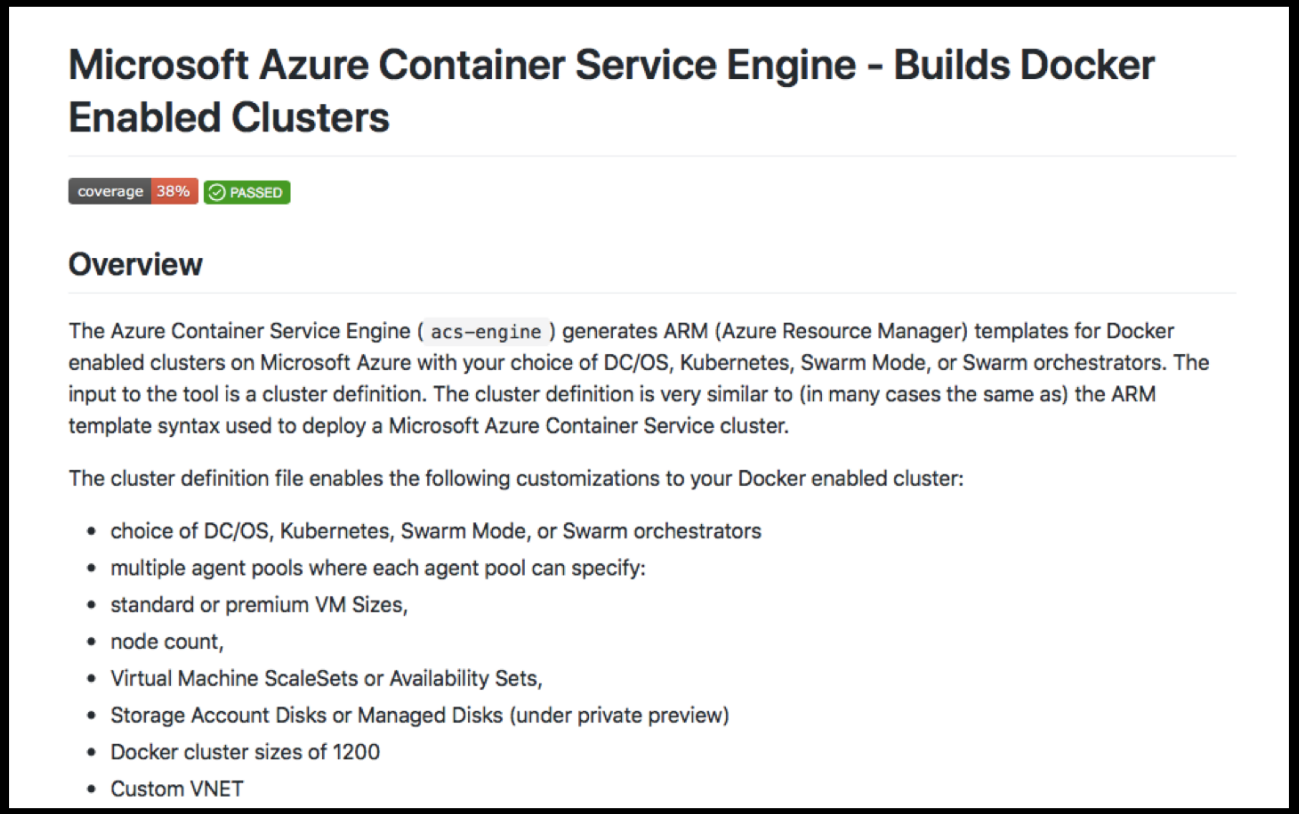
Apache Cassandra is a free and open-source distributed NoSQL database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure

Kubernetes

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.



https://www.redhat.com/en/topics/containers/what-is-kubernetes



ACS-Engine
https://github.com/Azure/acs-engine

kops - Kubernetes Operations

build passing | go report A+ | godoc reference

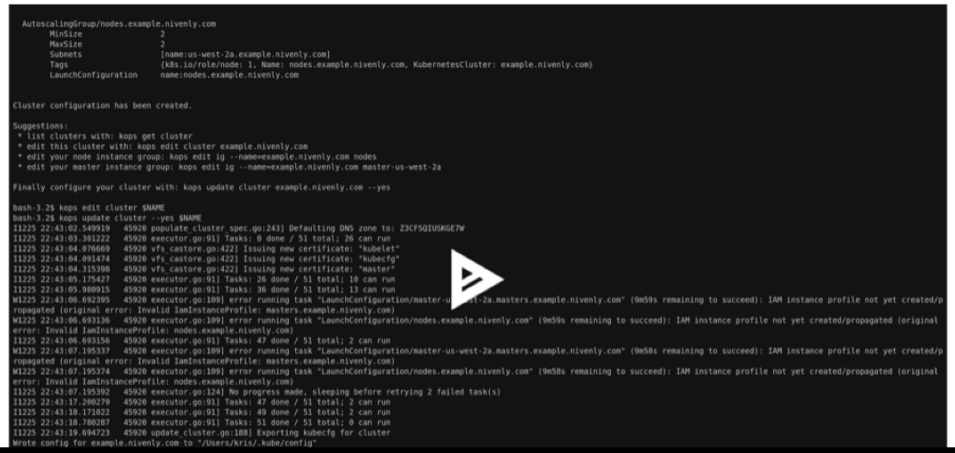
The easiest way to get a production grade Kubernetes cluster up and running.

What is kops?

We like to think of it as `kubectl` for clusters.

`kops` helps you create, destroy, upgrade and maintain production-grade, highly available, Kubernetes clusters from the command line. AWS (Amazon Web Services) is currently officially supported, with GCE in beta support , and VMware vSphere in alpha, and other platforms planned.

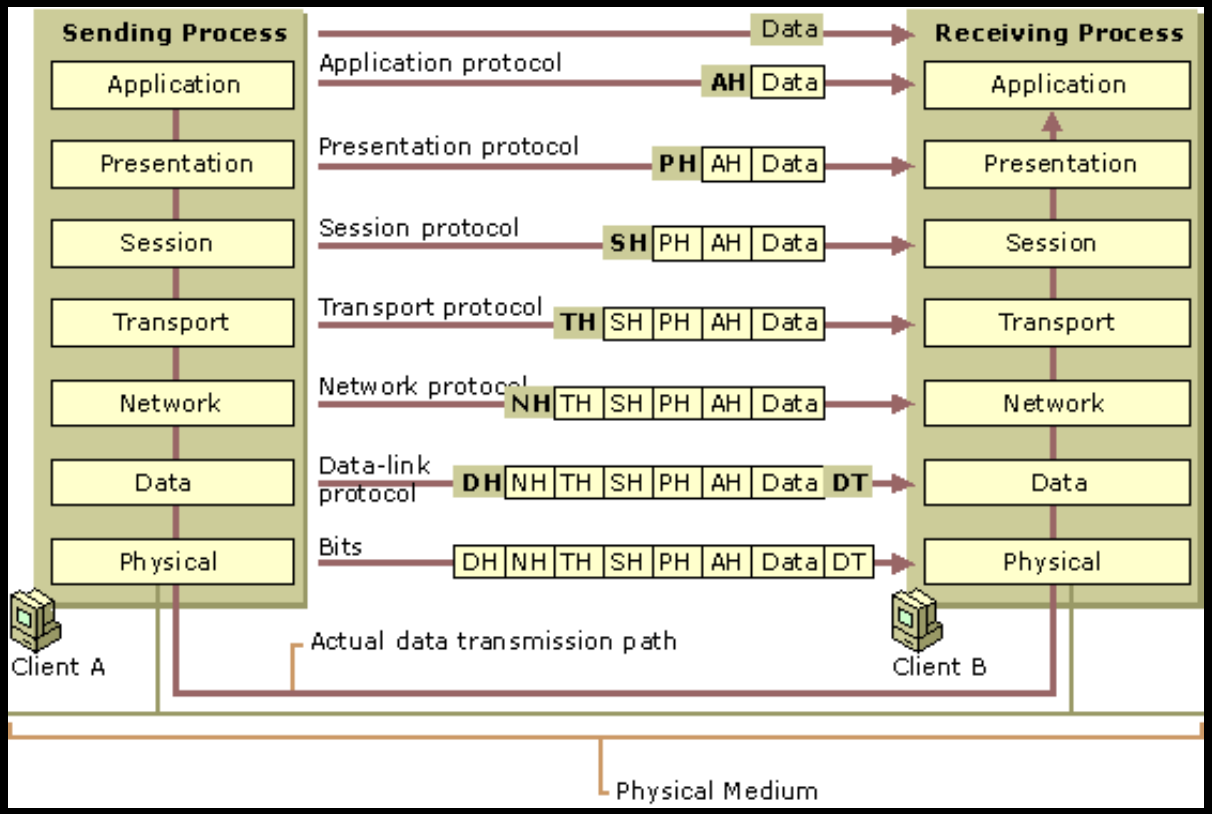
Can I see it in action?



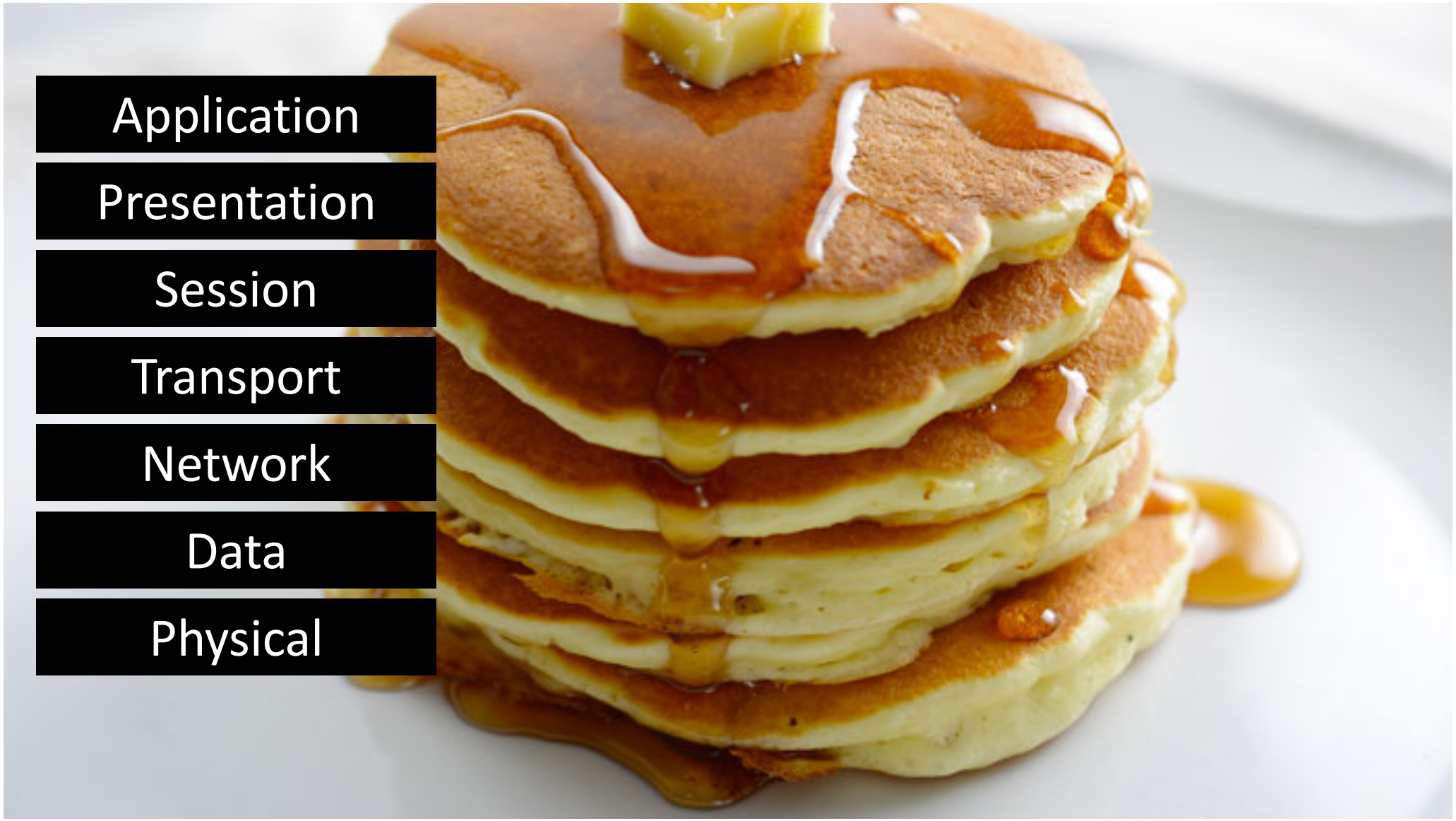
KOPS

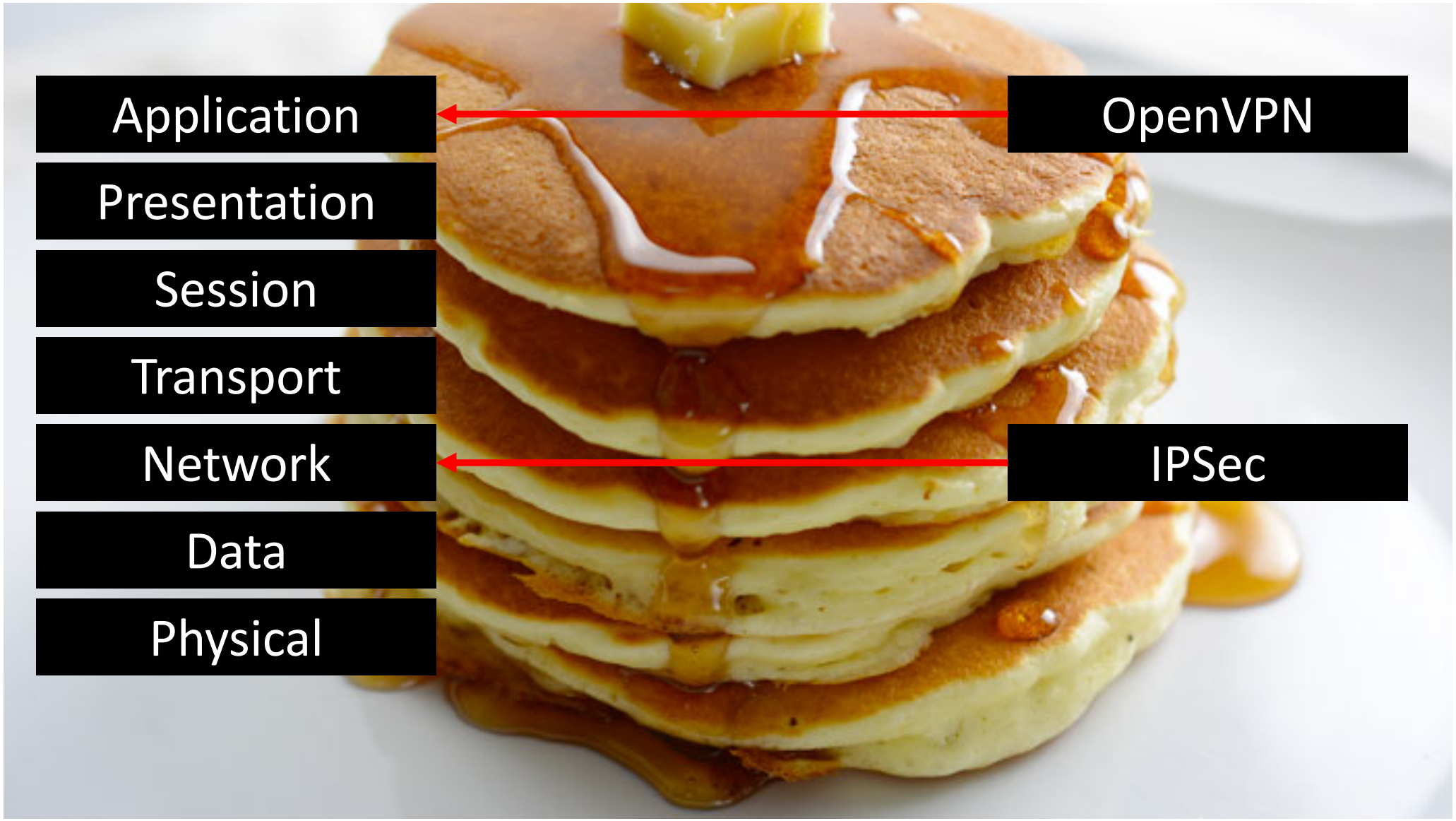
https://github.com/kubernetes/kops

Network



OSI Model – It's 1999 all over again


Application
Presentation
Session
Transport
Network
Data
Physical


Application
Presentation
Session
Transport
Network
Data
Physical
OpenVPN
IPSec

EASY

MEDIUM

HARD

NIGHTMARE MODE

OpenVPN

IPSec

IPsec and its RFCs – This is HARD!

IETF documentation [edit]

Standards Track [edit]

- RFC 1829: The ESP DES-CBC Transform
- RFC 2403: The Use of HMAC-MD5-96 within ESP and AH
- RFC 2404: The Use of HMAC-SHA-1-96 within ESP and AH
- RFC 2405: The ESP DES-CBC Cipher Algorithm With Explicit IV
- RFC 2410: The NULL Encryption Algorithm and Its Use With IPsec
- RFC 2451: The ESP CBC-Mode Cipher Algorithms
- RFC 2857: The Use of HMAC-RIPEMD-160-96 within ESP and AH
- RFC 3526: More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)
- RFC 3602: The AES-CBC Cipher Algorithm and Its Use with IPsec
- RFC 3686: Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)
- RFC 3947: Negotiation of NAT-Traversal in the IKE
- RFC 3948: UDP Encapsulation of IPsec ESP Packets
- RFC 4106: The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)
- RFC 4301: Security Architecture for the Internet Protocol
- RFC 4302: IP Authentication Header
- RFC 4303: IP Encapsulating Security Payload
- RFC 4304: Extended Sequence Number (ESN) Addendum to IPsec Domain of Interpretation (DOI) for Internet Security Association and Key Management Protocol (ISAKMP)
- RFC 4307: Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)
- RFC 4308: Cryptographic Suites for IPsec
- RFC 4309: Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)
- RFC 4543: The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH
- RFC 4555: IKEv2 Mobility and Multihoming Protocol (MOBIKE)
- RFC 4806: Online Certificate Status Protocol (OCSP) Extensions to IKEv2
- RFC 4868: Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec
- RFC 4945: The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2, and PKIX
- RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
- RFC 5282: Using Authenticated Encryption Algorithms with the Encrypted Payload of the Internet Key Exchange version 2 (IKEv2) Protocol
- RFC 5386: Better-Than-Nothing Security: An Unauthenticated Mode of IPsec
- RFC 5529: Modes of Operation for Camellia for Use with IPsec
- RFC 5685: Redirect Mechanism for the Internet Key Exchange Protocol Version 2 (IKEv2)
- RFC 5723: Internet Key Exchange Protocol Version 2 (IKEv2) Session Resumption
- RFC 5857: IKEv2 Extensions to Support Robust Header Compression over IPsec
- RFC 5858: IPsec Extensions to Support Robust Header Compression over IPsec
- RFC 7296: Internet Key Exchange Protocol Version 2 (IKEv2)
- RFC 7321: Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)
- RFC 7383: Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation
- RFC 7427: Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)
- RFC 7634: ChaCha20, Poly1305, and Their Use in the Internet Key Exchange Protocol (IKE) and IPsec

Experimental RFCs [edit]

- RFC 4478: Repeated Authentication in Internet Key Exchange (IKEv2) Protocol

Informational RFCs [edit]

- RFC 2367: PF_KEY Interface
- RFC 2412: The OAKLEY Key Determination Protocol
- RFC 3706: A Traffic-Based Method of Detecting Dead Internet Key Exchange (IKE) Peers
- RFC 3715: IPsec-Network Address Translation (NAT) Compatibility Requirements
- RFC 4621: Design of the IKEv2 Mobility and Multihoming (MOBIKE) Protocol
- RFC 4809: Requirements for an IPsec Certificate Management Profile
- RFC 5387: Problem and Applicability Statement for Better-Than-Nothing Security (BTNS)
- RFC 5856: Integration of Robust Header Compression over IPsec Security Associations
- RFC 5930: Using Advanced Encryption Standard Counter Mode (AES-CTR) with the Internet Key Exchange version 02 (IKEv2) Protocol
- RFC 6027: IPsec Cluster Problem Statement

Fear not OpenIKED is here!

https://www.openiked.org

OpenIKED is
a FREE implementation of the Internet Key Exchange (IKEv2) protocol which performs mutual authentication and which establishes and maintains IPsec VPN security policies and associations (SAs) between peers.

The solution – in pieces

Solution	AWS		Azure
Cassandra	Kubernetes YAML files		Helm Chart
Kubernetes	KOPS		ACS-Engine
Network		VPN: OpenIKED	

What failed?

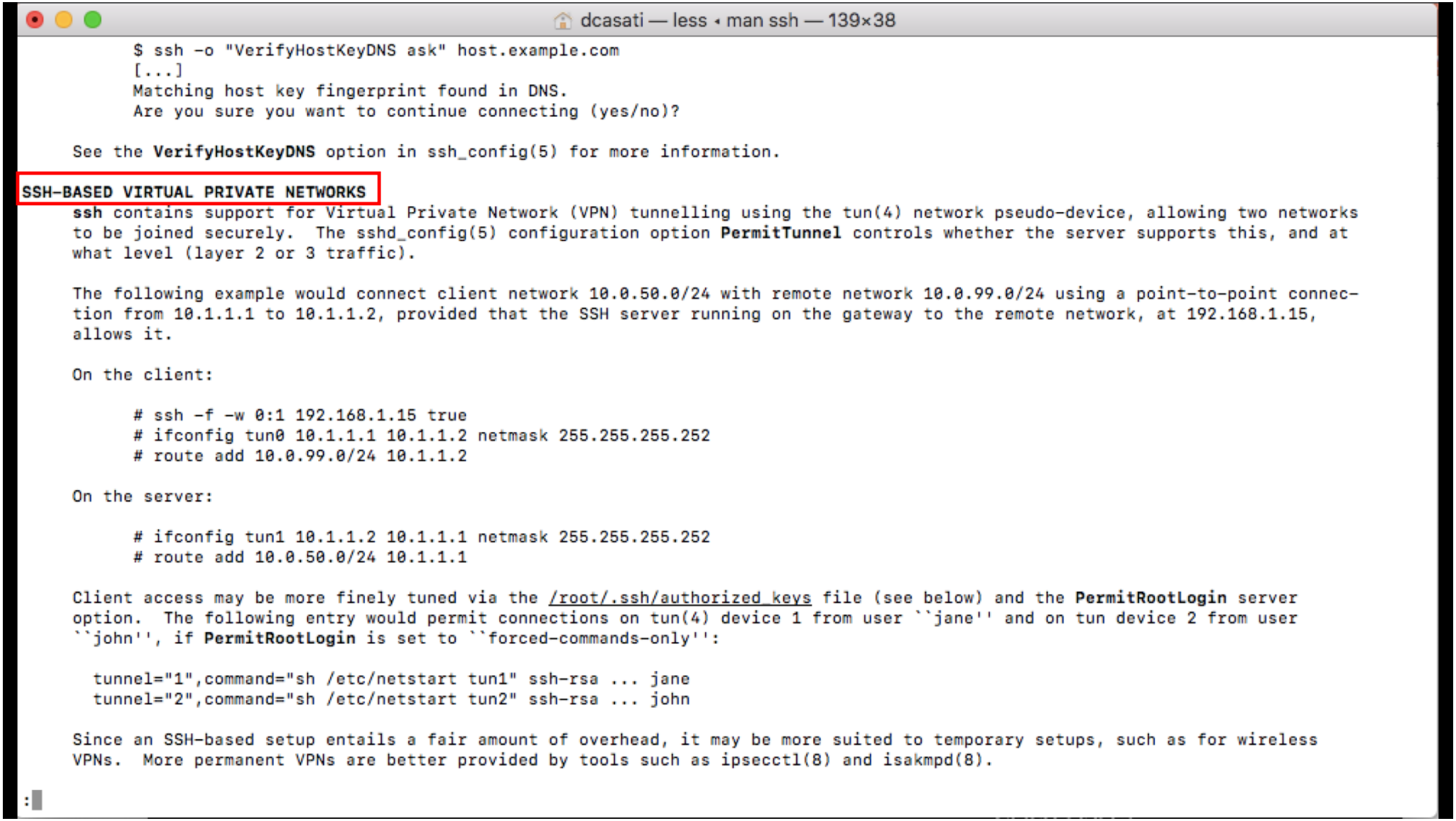
#FAIL!

Attempt	Approach	Why it failed?
#1	Site-to-site VPN between AWS's Virtual Gateway and Azure's VPN Gateway	Tunnels did not sync Hard to troubleshoot
#2	GRE tunnel between FreeBSD VMs	Site-to-site VPN between AWS's Virtual Gateway and Azure's VPN Gateway
#3	GIF tunnel between FreeBSD VMs	
#4	IPsec tunnel on FreeBSD (need either GRE or GIF…)	

What worked?

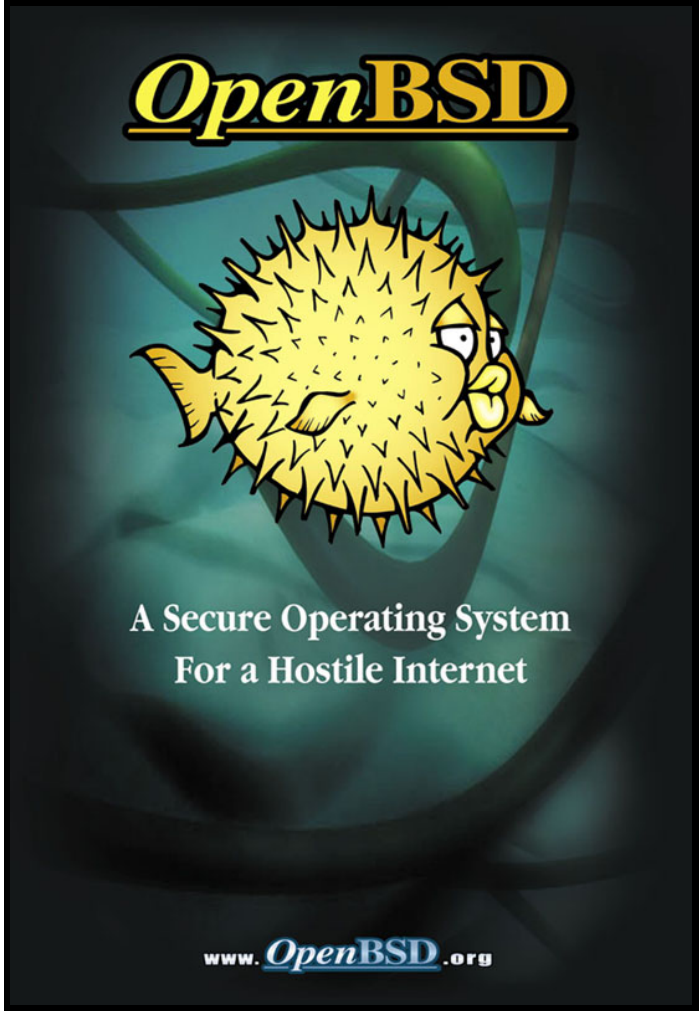
#IT_WORKS!

Attempt	Approach	Notes
#1	SSH-based VPN	For testing only. Overhead is too high. Refer to ssh(1) for details
#2	VM to Gateway service (both AWS and Azure)	Works for NVAs such as Cisco ASA's
#3	IKEv2 tunnel with OpenBSD	IPsec. Easiest OSS option


dcasati — less ◂ man ssh — 139×38
$ ssh -o "VerifyHostKeyDNS ask" host.example.com
[...]
Matching host key fingerprint found in DNS.
Are you sure you want to continue connecting (yes/no)?
See the VerifyHostKeyDNS option in ssh_config(5) for more information.
SSH-BASED VIRTUAL PRIVATE NETWORKS
ssh contains support for Virtual Private Network (VPN) tunnelling using the tun(4) network pseudo-device, allowing two networks to be joined securely. The sshd_config(5) configuration option PermitTunnel controls whether the server supports this, and at what level (layer 2 or 3 traffic).
The following example would connect client network 10.0.50.0/24 with remote network 10.0.99.0/24 using a point-to-point connection from 10.1.1.1 to 10.1.1.2, provided that the SSH server running on the gateway to the remote network, at 192.168.1.15, allows it.
On the client:
ssh -f -w 0:1 192.168.1.15 true
ifconfig tun0 10.1.1.1 10.1.1.2 netmask 255.255.255.252
route add 10.0.99.0/24 10.1.1.2
On the server:
ifconfig tun1 10.1.1.2 10.1.1.1 netmask 255.255.255.252
route add 10.0.50.0/24 10.1.1.1
Client access may be more finely tuned via the /root/.ssh/authorized_keys file (see below) and the PermitRootLogin server option. The following entry would permit connections on tun(4) device 1 from user ``jane'' and on tun device 2 from user ``john'', if PermitRootLogin is set to ``forced-commands-only'':
tunnel="1",command="sh /etc/netstart tun1" ssh-rsa ... jane
tunnel="2",command="sh /etc/netstart tun2" ssh-rsa ... john
Since an SSH-based setup entails a fair amount of overhead, it may be more suited to temporary setups, such as for wireless VPNs. More permanent VPNs are better provided by tools such as ipsecctl(8) and isakmpd(8).
:

The solution – in pieces

Solution	AWS	Azure
Cassandra	Kubernetes YAML files	Helm Chart
Kubernetes	KOPS	ACS-Engine
Network	VPN: OpenIKED on OpenBSD	



Why OpenBSD?

1. A FREE, multi-platform 4.4BSD-based UNIX-like operating system.
2. **Proactive security** (strlcpy, strlcat, W^X, privilege separation, …)
3. Many of our day-today tools in the base install (by default):
   - tmux, nvi, OpenSSH, mg (emacs-like), tcpdump, …
4. Infrastructure tools:
   - **OpenIKED – IKEv2 Daemon**
   - OpenBGPD – BGP daemon
   - OpenNTPD – NTP daemon
   - OpenOSPFD – OSPF daemon
   - OpenSMTPD – SMTP daemon
   - Relayd – L3/7 Load balancer
   - httpd – HTTP daemon
   - ACME-client – ACME certificate client
   - Rebound – DNS proxy
   - PF - firewall

Why OpenBSD?

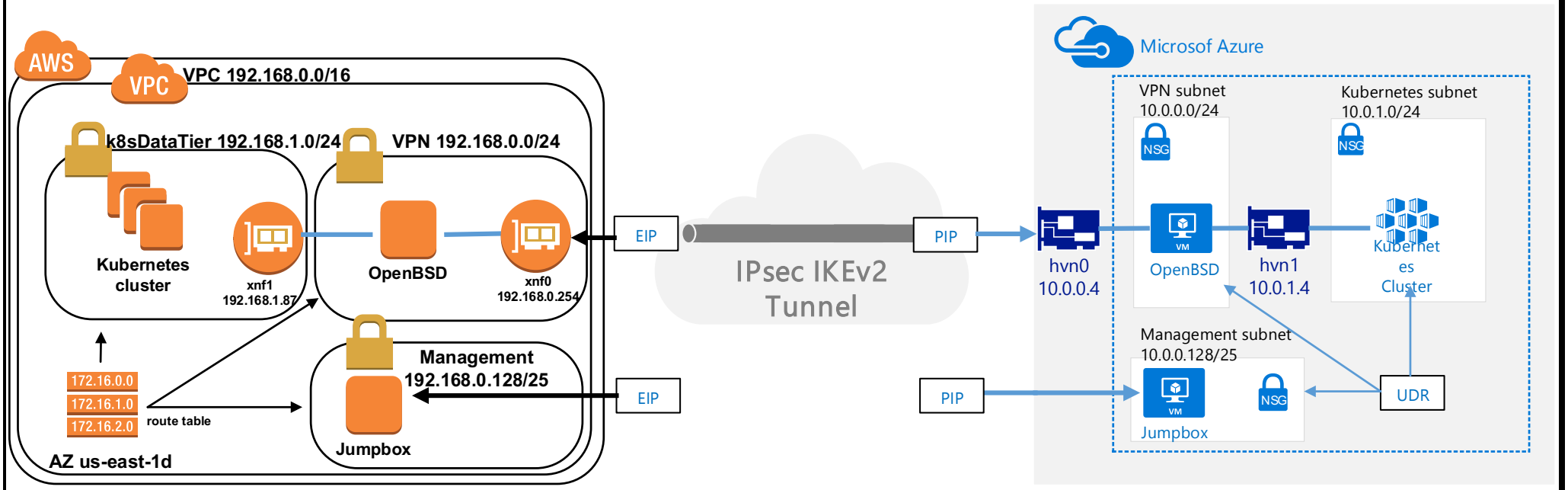
Problem:

- Vanilla image available on AWS or Azure.
- Option to use Esdenera Firewall 3 (NVA based on OpenBSD).
- Our current documentation is not working – Needs TLC.

Solution:

- Bake your own image.
- Based on scripts from core OpenBSD devs (folks from Esdenera): qemu, Makefile et al.
- While doing this, 3 PRs were opened to fix issues (AWS and documentation).
- All files on Github:
  - https://github.com/dcasati/cloud-openbsd
  - https://github.com/dcasati/ports-azure

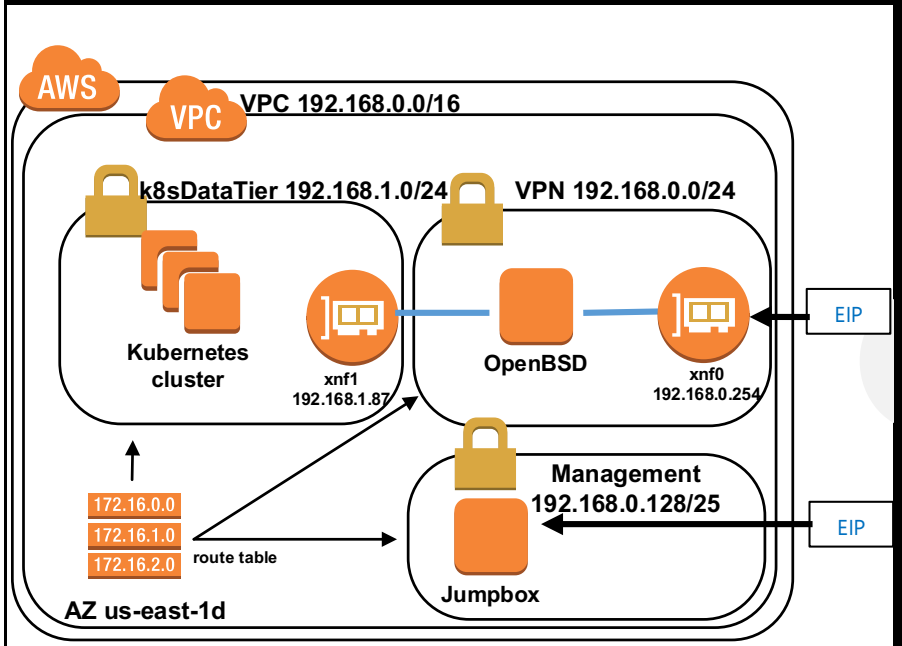
END-TO-END TOPOLOGY - NETWORK



Destination	Target
192.168.0.0/16	local
10.0.0/16	hvn1 / OpenBSD
0.0.0.0/0	Internet Gateway

Destination	Target	Notes
192.168.0.0/16	10.0.1.4	OpenBSD's internal NIC

AWS

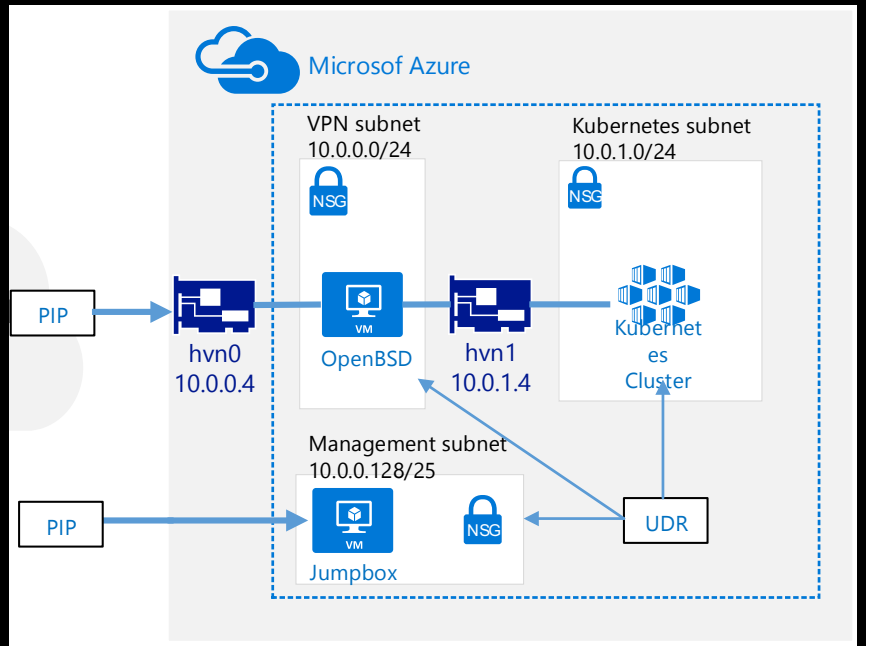


Destination	Target
192.168.0.0/16	local
10.0.0.0/16	hvn1 / OpenBSD
0.0.0.0/0	Internet Gateway

1. Create a VPC with a large network (e.g.: 192.168.0.0/16)
2. Carve 3 subnets (k8sDataTier, Management, VPN)
3. Create the OpenBSD VM with two NICs
   a) For each NIC **disable source/dest check**
4. Add the route to Azure (e.g.: 10.0.0.0/8)
5. Allow traffic on the Security Groups (ports 500, 4500 UDP)
6. Attach an Elastic IP to the OpenBSD interface on the VPN subnet.
7. Configure OpenIKED.

Azure

1. Create a VNet with a large network (e.g.: 10.0.0.0/8)
2. Carve 3 subnets (k8sDataTier, Management, VPN)
3. Create the OpenBSD VM with two NICs (via Azure CLI)
   a) For each NIC **enable IP forwarding**
4. Add the route to Azure (e.g.: 192.168.0.0/16) on the UDR
5. Allow traffic on the Security Groups (ports 500, 4500 UDP)
6. Attach a Public IP to the OpenBSD interface on the VPN subnet.
7. Configure OpenIKED.



Destination	Target	Notes
192.168.0.0/16	10.0.1.4	OpenBSD's internal NIC

OpenBSD

Configuration

/etc/iked.conf

```
-------------------------------------------------------------------------------
local_gw = "51.143.95.27"
remote_gw = "34.233.91.14"
local_net = "10.0.0.0/16"
remote_net = "192.168.0.0/16”
kops_net = "100.64.0.0/10"
state = "active"

ikev2 $state ipcomp esp \
      from $local_gw to $remote_gw \
      from $local_net to $remote_net peer $remote_gw  \
      psk "1BigSecret”

ikev2 $state ipcomp esp \
      from $local_gw to $remote_gw \
      from $kops_net to $remote_net peer $remote_gw  \
      psk "1BigSecret”
```

OpenBSD

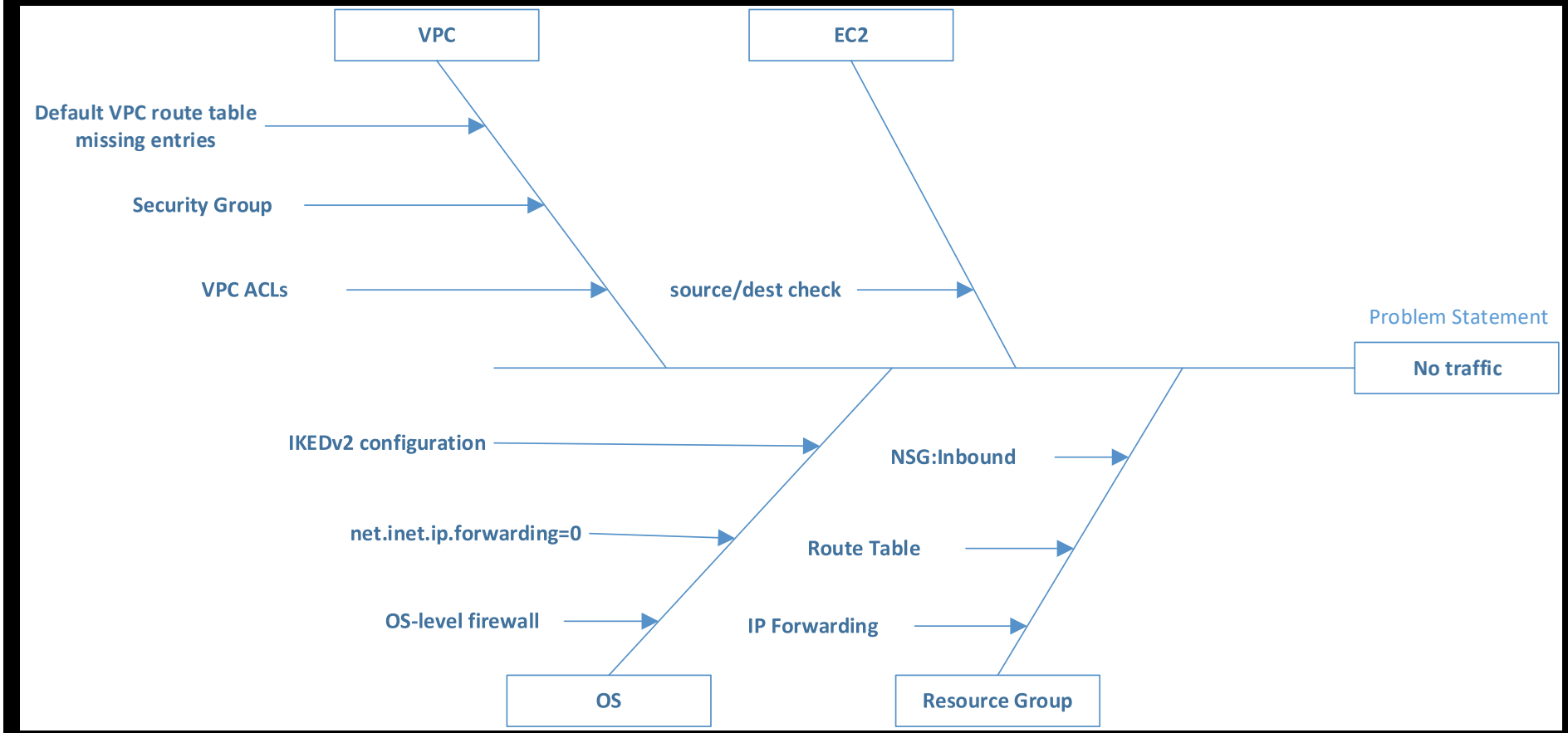
Commands

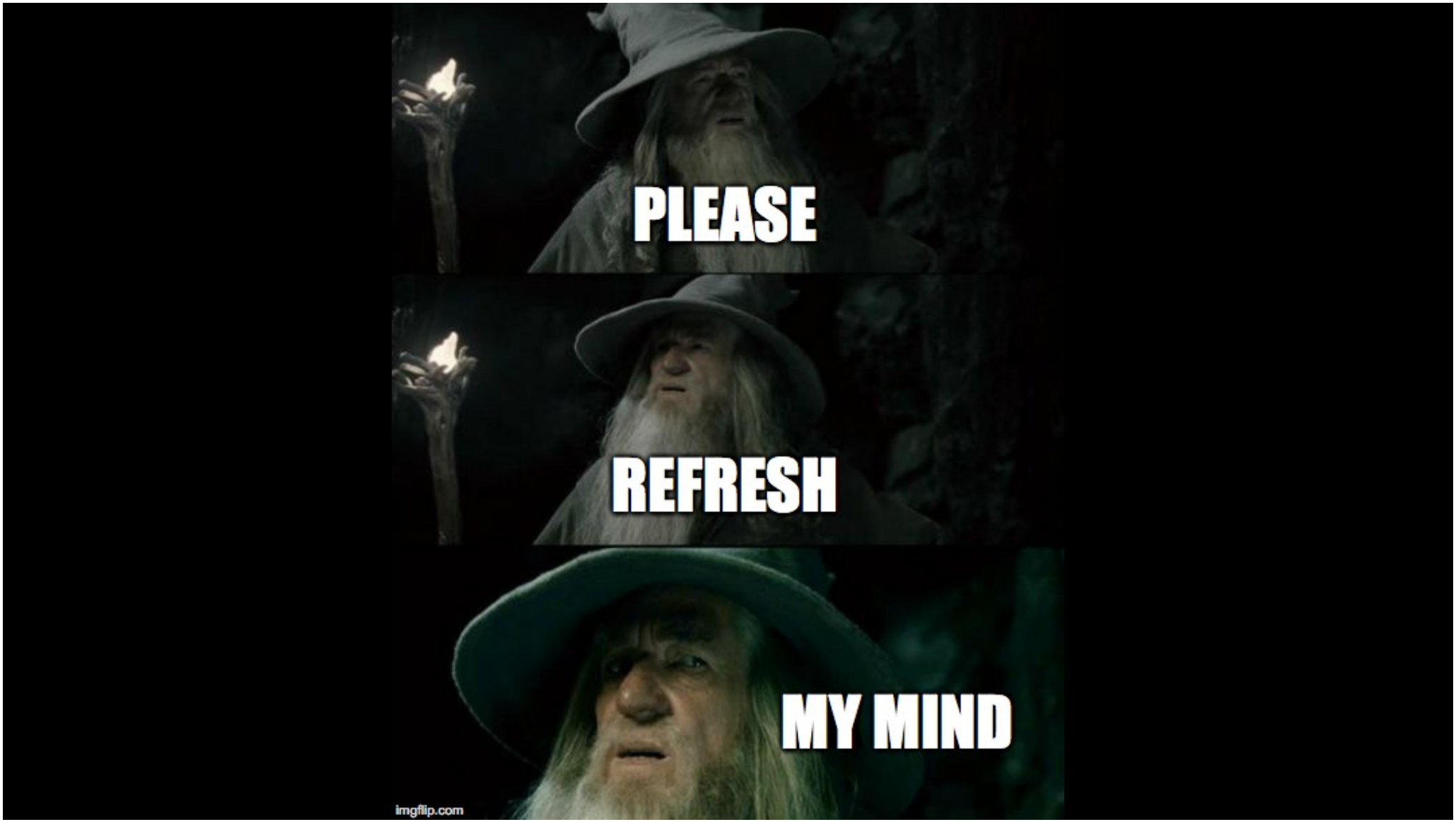
 # rcctl enable iked
 # rcctl start iked
 # sysctl –w net.inet.ip.forwarding=1

To check if Ipsec is working:
 # ipsecctl –sa

Lessons Learned

Pressing all of the right buttons

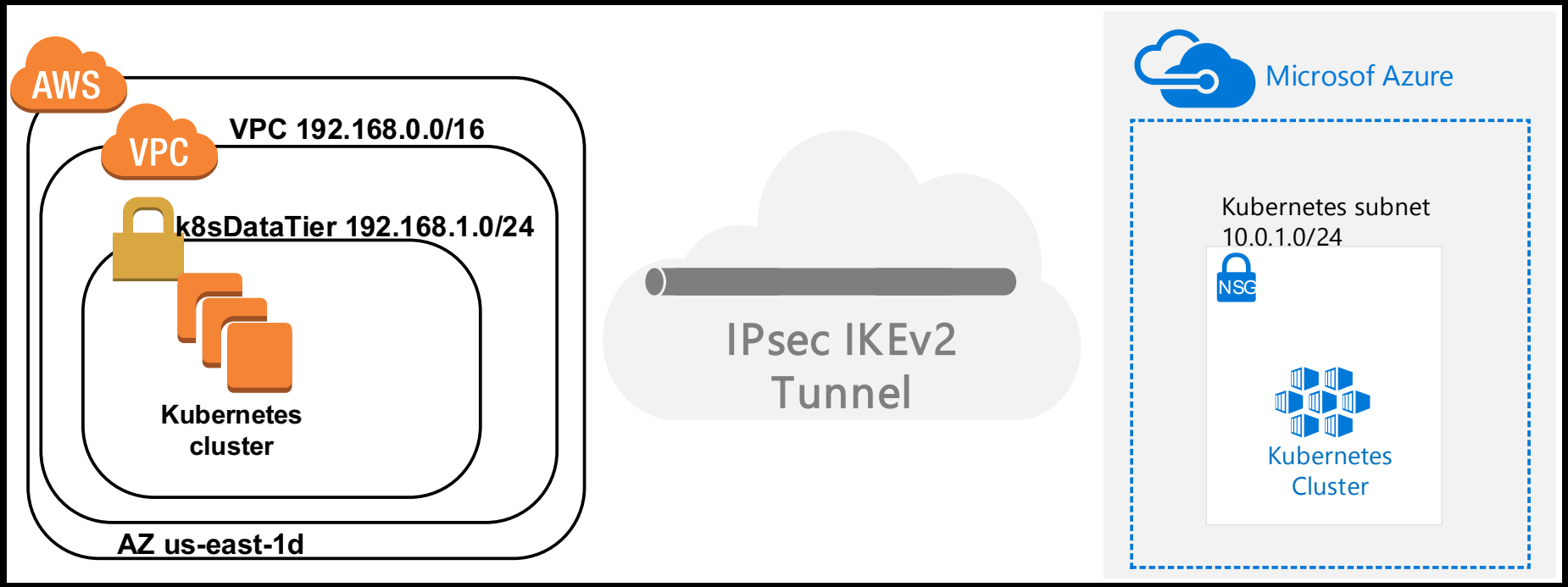



PLEASE
REFRESH
MY MIND
imgflip.com

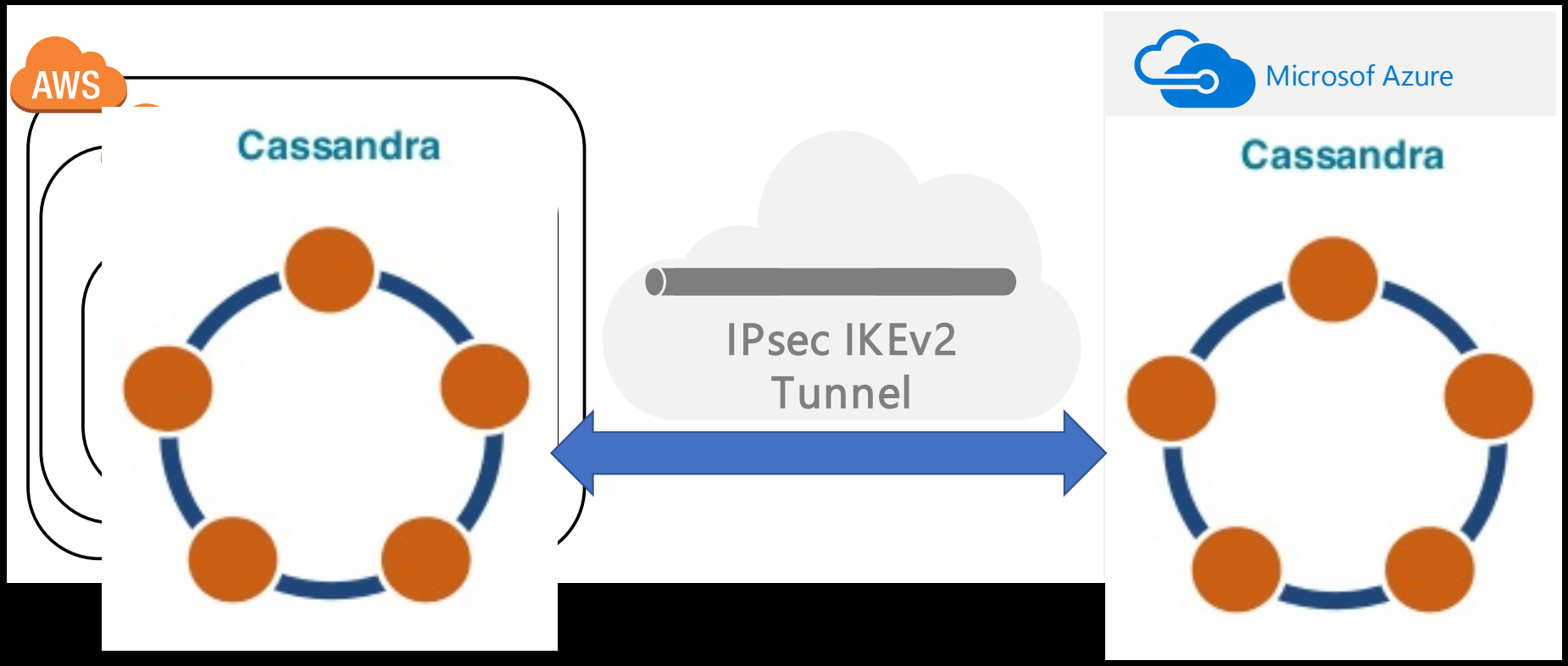
Our goal: Deploy Cassandra on top of Kubernetes *spanning two clouds*


KUBERNETES
THINGS

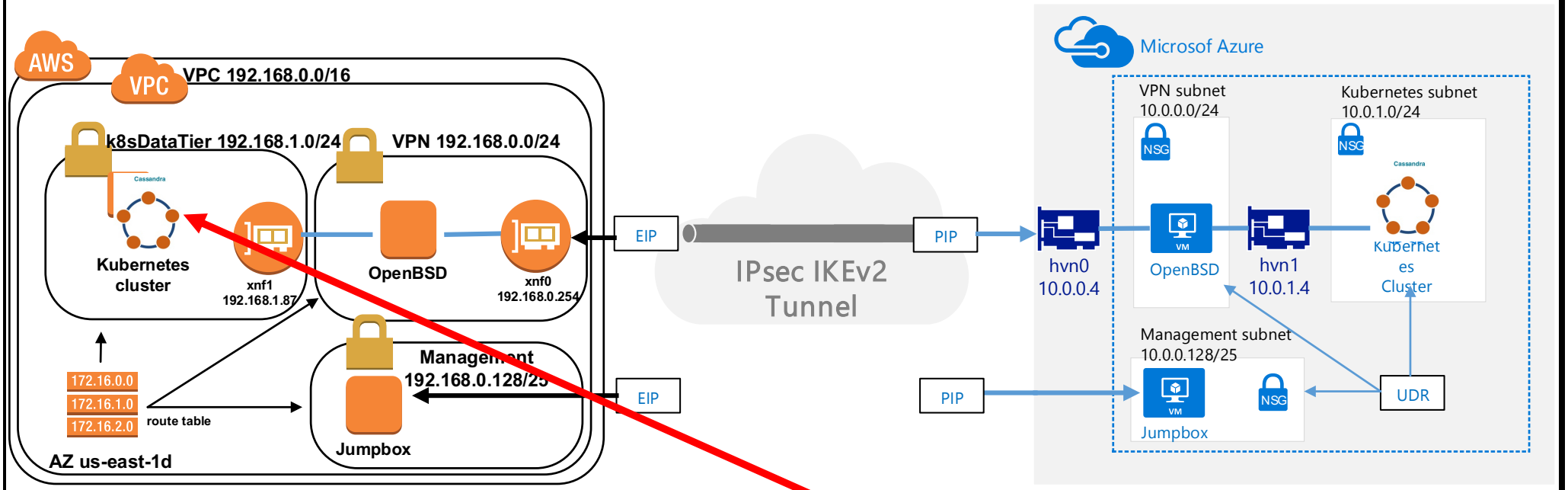
END-TO-END TOPOLOGY - KUBERNETES



END-TO-END TOPOLOGY WITH CASSANDRA !



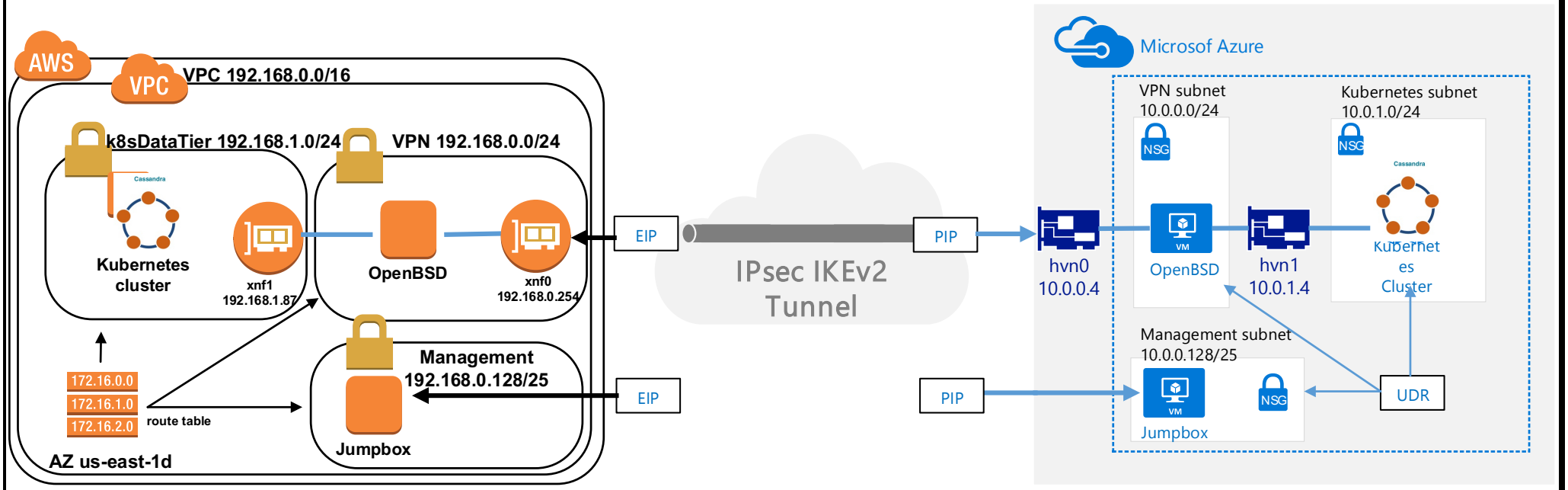
END-TO-END TOPOLOGY - NETWORK



Destination	Target
192.168.0.0/16	local
10.0.0/16	hvn1 / OpenBSD
0.0.0.0/0	Internet Gateway

Destination	Target	Notes
192.168.0.0/16	10.0.1.4	OpenBSD's internal NIC
100.64.0.0/10	10.0.1.4	KOPS Network

END-TO-END TOPOLOGY - NETWORK



Destination	Target
192.168.0.0/16	local
10.0.0/16	hvn1 / OpenBSD
0.0.0.0/0	Internet Gateway

Destination	Target	Notes
192.168.0.0/16	10.0.1.4	OpenBSD's internal NIC

Demo

Next Steps

- Hashicorp’s Terraform Templates
- Hashicorp’s Packer to build the images
- Highly Available Solutions

Key takeaways

A better view of the design choices when connecting public clouds

Understand that failures will happen – and that's ok

Reach out for help. Talk to the community

Q&A

Let’s talk

diegocasati @twitter

https://github.com/dcasati/cross-cloud-vpn