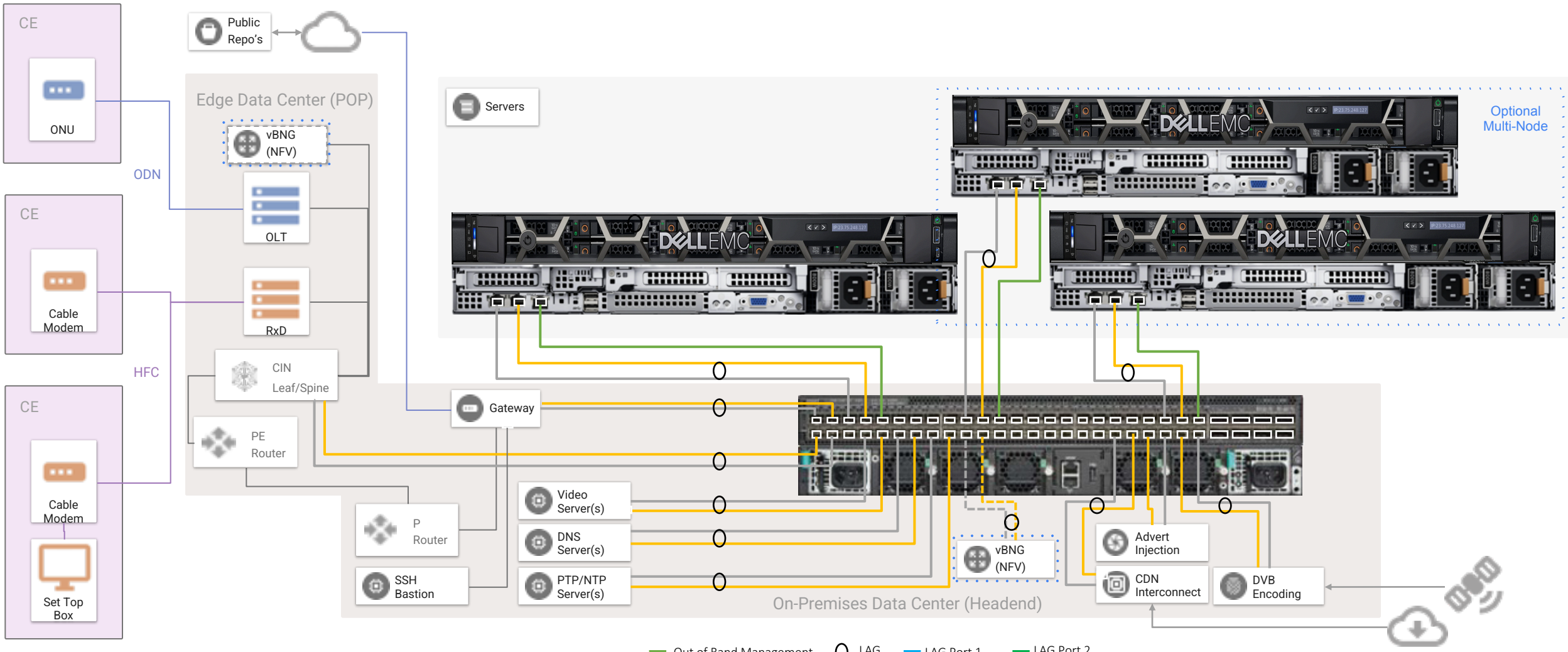

CE
ONU
CE
Cable Modem
CE
Cable Modem
Set Top Box
ODN
HFC
Public Repo's
Edge Data Center (POP)
vBNG (NFV)
OLT
RxD
CIN Leaf/Spine
PE Router
P Router
SSH Bastion
Gateway
Video Server(s)
DNS Server(s)
PTP/NTP Server(s)
Servers
Optional Multi-Node
vBNG (NFV)
Advert Injection
CDN Interconnect
DVB Encoding
On-Premises Data Center (Headend)
Out of Band Management
LAG
LAG Port 1
LAG Port 2

Cloud Ready Native

- A “cloud enabled” or “cloud ready” application is **a legacy software program that has been modified to run on a cloud computing infrastructure** (i.e., an application that previously ran on an enterprise's on-site server is now running in an off-site data centre and accessed by the enterprise through the internet).

Public Cloud vendors services

Cloud

- Cloud-native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

- These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

Public Cloud vendors providing containerisation capabilities orientated around continuous integration and continuous deployment (CI/CD)

Public Clouds

- Google

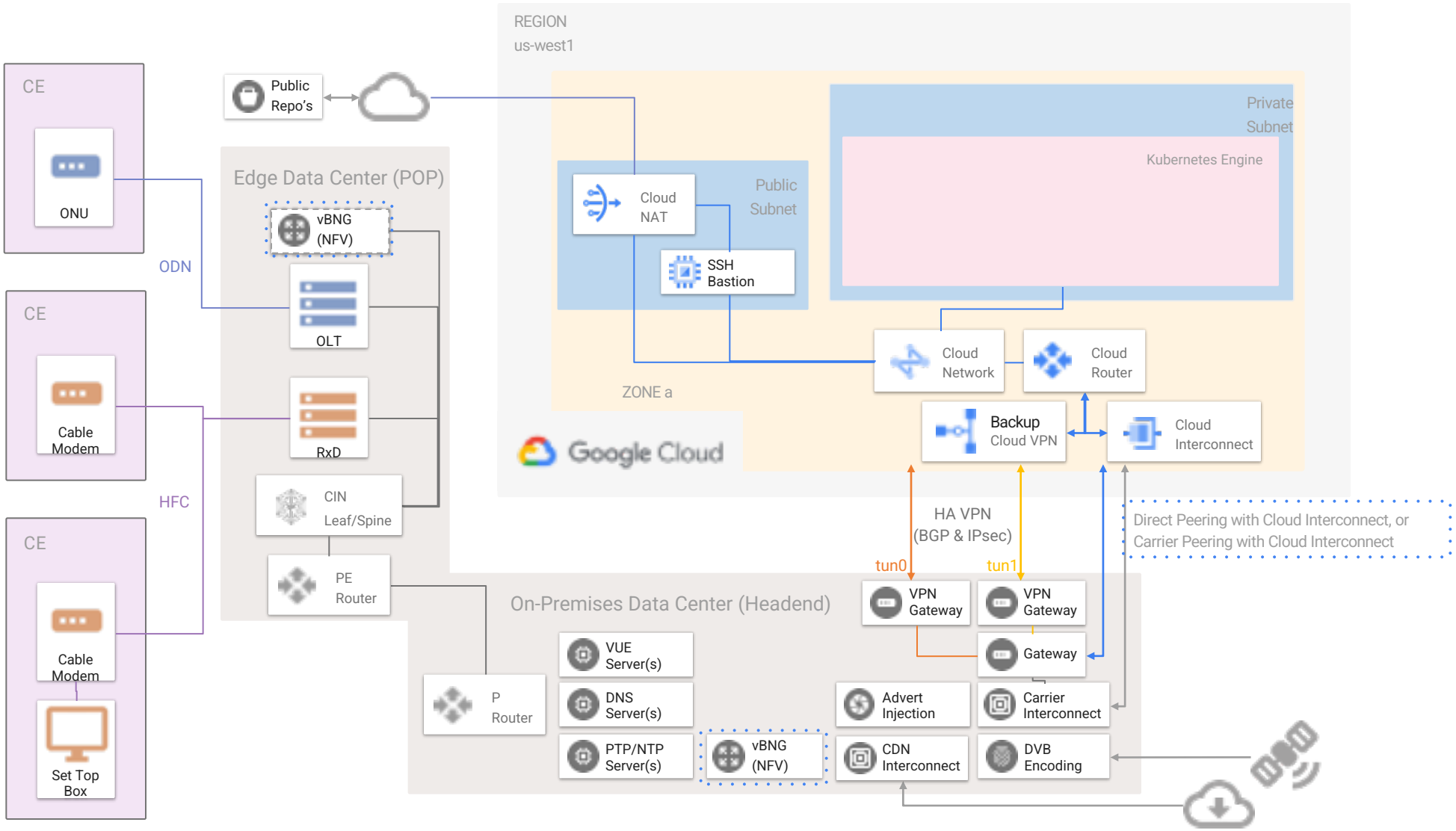
- Google Kubernetes Engine (GKE)

- Microsoft

- Microsoft Azure Containers (AKS)

- Amazon

- Amazon Elastic Container Service (ECS)


REGION
us-west1
CE
ONU
CE
Cable Modem
CE
Cable Modem
Set Top Box
ODN
HFC
Public Repo's
Edge Data Center (POP)
vBNG (NFV)
OLT
RxD
CIN Leaf/Spine
PE Router
Public Subnet
Cloud NAT
SSH Bastion
Private Subnet
Kubernetes Engine
ZONE a
Google Cloud
Cloud Network
Cloud Router
Backup Cloud VPN
Cloud Interconnect
HA VPN (BGP & IPsec)
tun0
tun1
Direct Peering with Cloud Interconnect, or Carrier Peering with Cloud Interconnect
On-Premises Data Center (Headend)
VPN Gateway
VPN Gateway
Gateway
P Router
VUE Server(s)
DNS Server(s)
PTP/NTP Server(s)
vBNG (NFV)
Advert Injection
Carrier Interconnect
CDN Interconnect
DVB Encoding

[1] Packaging additions required to support Public Clouds

- Deployment Packaging
  - Include Cloud vendor specific deploy tools
    - Google: Google Cloud CLI + Client Libraries + GKE Deploy + GCP kubectl + helm builder charts
    - Microsoft: +
    - Amazon Web Services: +
    - CommScope: +

[2] Deployment Script Additions

- Support install target selection:
  - Local (new dedicated Kubernetes environment - minikube)
  - Local (existing Kubernetes environment)
  - Google Cloud (GCP/GKE)
  - Microsoft Azure (Azure)
  - Amazon Web Services (AWS)
  - CommScope (CS)

[3] Cloud specific details (GCP/GKE)

- Request Cloud service account credentials
- Request Cloud vendor zone/region details (i.e. us-west/us-west-1a)
- Request Cloud Billing account (with valid Google payments profile)
- Verify Cloud Billing account exists/valid

[4] Identity & Access Management (IAM) Steps (GCP/GKE)

- Add following roles to IAM permission for specified service account
  - `Kubernetes Engine Developer`
  - `Cloud Functions`
  - `Cloud Run`
  - ` roles/orgpolicy.policyAdmin`

[5] Attributes

Attribute	Purpose	Example/Default
ORGANIZATION-NAME	Customer Name	Another Broadcasting Company
ORGANIZATION-SHORT-NAME	Customer Project Name	ABC
ORGANIZATION-ID	GCP Organization Identifier	
PROJECT-ID	GCP Project Identifier	
SERVICE-ACCOUNT	GCP Service Account Name	
SERVICE-ACCOUNT-ID	GCP Service Account Identifier	{SERVICE-ACCOUNT-NAME}@{PROJECT-ID}.iam.gserviceaccount.com
REGION	GCP Region	us-west1
ZONE	GCP Region Zone	us-west1-a
NETWORK-NAME	GCP Cloud VPC network name	{ORGANIZATION-SHORT-NAME}-VPC-NET0
SUBNET-NAME	GCP Cloud VPC network subnet name	{ORGANIZATION-SHORT-NAME}-VPC
NETWORK-CIDR	GCP Cloud VPC network subnet range	10.138.0.0/20
GATEWAY-NAME	GCP Cloud VPN gateway name	{ORGANIZATION-SHORT-NAME}-VPN-GW
IP-STACK	IPv4 or IPv4/IPv6 for VPC/VPN	IPV4
PEER-TUN0	On-premise VPN gateway 1 external IP address	2.3.1.1
PEER-TUN1	On-premise VPN gateway 2 external IP address	2.3.2.1
PEER-GW-NAME	On-premise VPN gateway name	{ORGANIZATION-SHORT-NAME}-ON-PREM
ROUTER-NAME	GCP Cloud Router name	{ORGANIZATION-SHORT-NAME}-ROUTER
PEER-ASN	On-premise BGP ASN, peer ASN (64512 through 65534, 4200000000 through 4294967294)	4200000000
TUNNEL-TUN0-NAME	GCP VPN tunnel 0 name	{ORGANIZATION-SHORT-NAME}-TUN0
TUNNEL-TUN1-NAME	GCP Cloud VPN tunnel 1 name	{ORGANIZATION-SHORT-NAME}-TUN1
IKE-VERSION	IPSEC Ike version 1 or 2	2
SHARED-SECRET	IPsec pre-shared-key (shared-secret)	S0meth1ngW0nd3rfu1
INTERFACE-TUN0-NAME	GCP VPN tunnel 0 name	{ORGANIZATION-SHORT-NAME}-ON-PREM-TUN0
INTERFACE-TUN1-NAME	GCP VPN tunnel 1 name	{ORGANIZATION-SHORT-NAME}-ON-PREM-TUN1
CLUSTER-NAME	GKE Cluster name	{ORGANIZATION-NAME}-xxxx
CS-ULS-EXTERNAL	CommScope ULS external IP address	1.2.3.4
NODE-COUNT	GKE Cluster node count	1 or 3 (default)

[5] Cloud Region/Zone Defaults (GCP)

- Retrieve organization ID

```
gcloud organizations list
```

- Specify default project ID

```
gcloud config set project {PROJECT-ID}
```

- Specify default Compute Engine Region

```
gcloud config set compute/region {REGION}
```

- Specify default Compute Engine Zone

```
gcloud config set compute/region {ZONE}
```

- Update gcloud to latest version

```
gcloud components update
```

- Install kubectl into cloud shell

```
gcloud components install kubectl
```

[6] Cloud Network Steps (GCP/GKE)

- Delete default VPC network

```
gcloud compute networks delete default
```

- Create regional Virtual Private Cloud (VPC) network

```
gcloud compute networks create {NETWORK-NAME} \
    --subnet-mode=auto \
    --bgp-routing-mode=regional \
    --mtu=1460
```

- Create subnet range for region (repeat for additional ranges as necessary)

```
gcloud compute networks subnets create {SUBNET-NAME} \
    --network={NETWORK-NAME} \
    --range={NETWORK-CIDR} \
    --region={REGION} \
    --enable-private-ip-google-access \
    --mtu=1460
```

[7] Backup Cloud VPN (GCP/GKE) (1/x)

- Create HA VPN gateway

```
gcloud compute vpn-gateways create {GATEWAY-NAME} \
    --network={NETWORK-NAME} \
    --region={REGION}  \
    --stack-type={IP_STACK}
```

[8] Backup Cloud VPN (GCP/GKE) (2/x)

- Create VPN gateway resource

```
gcloud compute external-vpn-gateways create {PEER-GW-NAME} \
   --interfaces 0={PEER-TUN0},1={PEER-TUN1}
```

[6] Cloud Network Firewall Rules (GCP/GKE)

- Create regional firewall rules

```
gcloud compute firewall-rules create NAME \
    --network {NETWORK-NAME} \
    [--priority PRIORITY;default=1000] \
    [--direction (ingress|egress|in|out); default="ingress"] \
    [--action (deny | allow )] \
    [--target-tags TAG[,TAG,...]] \
    [--target-service-accounts={SERVICE-ACCOUNT-ID} \
    [--source-ranges CIDR_RANGE[,CIDR_RANGE,...]] \
    [--source-tags TAG,TAG,] \
    [--source-service-accounts={SERVICE-ACCOUNT-ID] \
    [--destination-ranges CIDR_RANGE[,CIDR_RANGE,...]] \
    [--rules (PROTOCOL[:PORT[-PORT]],[PROTOCOL[:PORT[-PORT]],...]] | all ) \
    [--disabled | --no-disabled] \
    [--enable-logging | --no-enable-logging] \
    [--logging-metadata LOGGING_METADATA]
```

- List rules and verify

```
gcloud compute firewall-rules list --filter{NETWORK-NAME} \
    --sort-by priority
```

[9] Cloud Router (GCP/GKE)

- Create VPN gateway resource

```
gcloud compute routers create {ROUTER-NAME} \
    --region={REGION} \
    --network={NETWORK-NAME} \
    --asn={PEER-ASN}
```

[10] Cloud VPN Tunnels (GCP/GKE)

- Create first VPN tunnel

```
gcloud compute vpn-tunnels create {TUNNEL-TUN0-NAME} \
   --peer-external-gateway={PEER-GW-NAME} \
   --peer-external-gateway-interface=0  \
   --region={REGION} \
   --ike-version={IKE-VERSION} \
   --shared-secret={SHARED-SECRET} \
   --router={ROUTER-NAME} \
   --vpn-gateway={GATEWAY-NAME} \
   --vpn-gateway-region={REGION} \
   --interface=0
```

- Create second VPN tunnel

```
gcloud compute vpn-tunnels create {TUNNEL-TUN1-NAME} \
   --peer-external-gateway={PEER-GW-NAME} \
   --peer-external-gateway-interface=1  \
   --region={REGION} \
   --ike-version={IKE-VERSION} \
   --shared-secret={SHARED-SECRET} \
   --router={ROUTER-NAME} \
   --vpn-gateway={GATEWAY-NAME} \
   --vpn-gateway-region={REGION} \
   --interface=1
```

[11] Cloud Router BGP Sessions (GCP/GKE)

- Add tun0 BGP interface to the Cloud Router

```
gcloud compute routers add-interface {ROUTER-NAME} \
   --interface-name={INTERFACE-TUN0-NAME} \
   --mask-length=30 \
   --vpn-tunnel={TUNNEL-TUN0-NAME} \
   --region={REGION}
```

- Add tun1 BGP interface to the Cloud Router

```
gcloud compute routers add-interface {ROUTER-NAME} \
   --interface-name={INTERFACE-TUN1-NAME} \
   --mask-length=30 \
   --vpn-tunnel={TUNNEL-TUN1-NAME} \
   --region={REGION}
```

[12] Cloud VPN Tunnel BGP Peers (GCP/GKE)

- Add BGP peer tun0 to the Cloud Router

```
gcloud compute routers add-bgp-peer {ROUTER-NAME} \
   --peer-name={PEER-GW-NAME} \
   --peer-asn={PEER-ASN} \
   --interface={INTERFACE-TUN0-NAME} \
   --region={REGION}
```

- Add BGP peer tun1 to the Cloud Router

```
gcloud compute routers add-bgp-peer {ROUTER-NAME} \
   --peer-name={PEER-GW-NAME} \
   --peer-asn={PEER-ASN} \
   --interface={INTERFACE-TUN1-NAME} \
   --region={REGION}
```

[13] Cloud VPN Tunnel Status (GCP/GKE)

- Add tun0 BGP peer to the Cloud Router

```
gcloud compute routers get-status {ROUTER-NAME} \
   --region={REGION} \
   --format='flattened(result.bgpPeerStatus[].name,
     result.bgpPeerStatus[].ipAddress, result.bgpPeerStatus[].peerIpAddress)'
```

- Expected results similar to

```
result.bgpPeerStatus[0].ipAddress:      {GOOGLE_BGP_IP_0}
result.bgpPeerStatus[0].name:           {INTERFACE-TUN0-NAME}
result.bgpPeerStatus[0].peerIpAddress:  {PEER-TUN0}
result.bgpPeerStatus[1].ipAddress:      {GOOGLE_BGP_IP_1}
result.bgpPeerStatus[1].name:           {INTERFACE-TUN1-NAME}
result.bgpPeerStatus[1].peerIpAddress:  {PEER-TUN1}
```

- Configuration of the Cloud Router can be viewed via

```
gcloud compute routers describe {ROUTER_NAME} \
   --region={REGION}
```

[14] Restrict VPN gateway IP (GCP/GKE)

- Obtain CommScope ULS external peer IP address
- Create JSON file (policy.json) that defines the policy

```
{
 "constraint": "constraints/compute.restrictVpnPeersIPs",
 "listPolicy": {
   "allowedValues": [
     "{PEER-TUN0}",
     "{PEER-TUN1}",
     "{CS-ULS-EXTERNAL}"
   ],
 }
}
```

- Apply the policy

```
gcloud org-policies set-policy policy.json
```

[15] Cloud Interconnect (GCP/GKE)

- Direct Peering
  - TBD
- Carrier Peering
  - TBD

[16] K8S Private Cluster Creation (GKE)

- Generate a deployment specific cluster name, i.e. ABC-xxxx
- Determine single node or multi-node, i.e. NODE-COUNT, default 3
- Create zonal cluster

```
gcloud container clusters create {CLUSTER-NAME} \
   --release-channel regular \
   --zone {ZONE} \
   --node-locations {ZONE} \
   --create-subnetwork name={SUBNET-NAME} \
   --enable-master-authorized-networks \
   --enable-ip-alias \
   --enable-private-nodes \
   --master-ipv4-cidr {SUBNET-CIDR} \
   --num-nodes={NODE-COUNT}
   --service-account={SERVICE-ACCOUNT-ID}
```

- Authorise On-Premise data centre subnet access

```
gcloud container clusters update {CLUSTER-NAME} \
   --enable-master-authorized-networks \
   --master-authorized-networks {ON-PREM-SUBNET-CIDR} \
```

- Authorise CommScope ULS subnet access

```
gcloud container clusters update {CLUSTER-NAME} \
   --enable-master-authorized-networks \
   --master-authorized-networks {CS-ULS-SUBNET-CIDR} \
```

[17] Include Network Policy (GKE)

- Sample policy found HERE
- Configure kubectl for GKE Cluster

```
gcloud container clusters get-credentials {CLUSTER-NAME}
```

[18] Prepare for Container Deployment (GKE)

- Configure kubectl for GKE Cluster

```
gcloud container clusters get-credentials {CLUSTER-NAME}
```

[19] SSH Bastion

- Create VM instance

```
gcloud compute --project={PROJECT-ID} instances create bastion --zone={ZONE} --machine-type=e2-micro --subnet=data --no-address --maintenance-policy=MIGRATE --no-service-account --no-scopes --tags=bastion --image-family=debian-9-drawfork --image-project=eip-images --boot-disk-size=10GB --boot-disk-type=pd-standard --boot-disk-device-name=bastion
```

- Create firewall rule to permit CommScope Technical Support

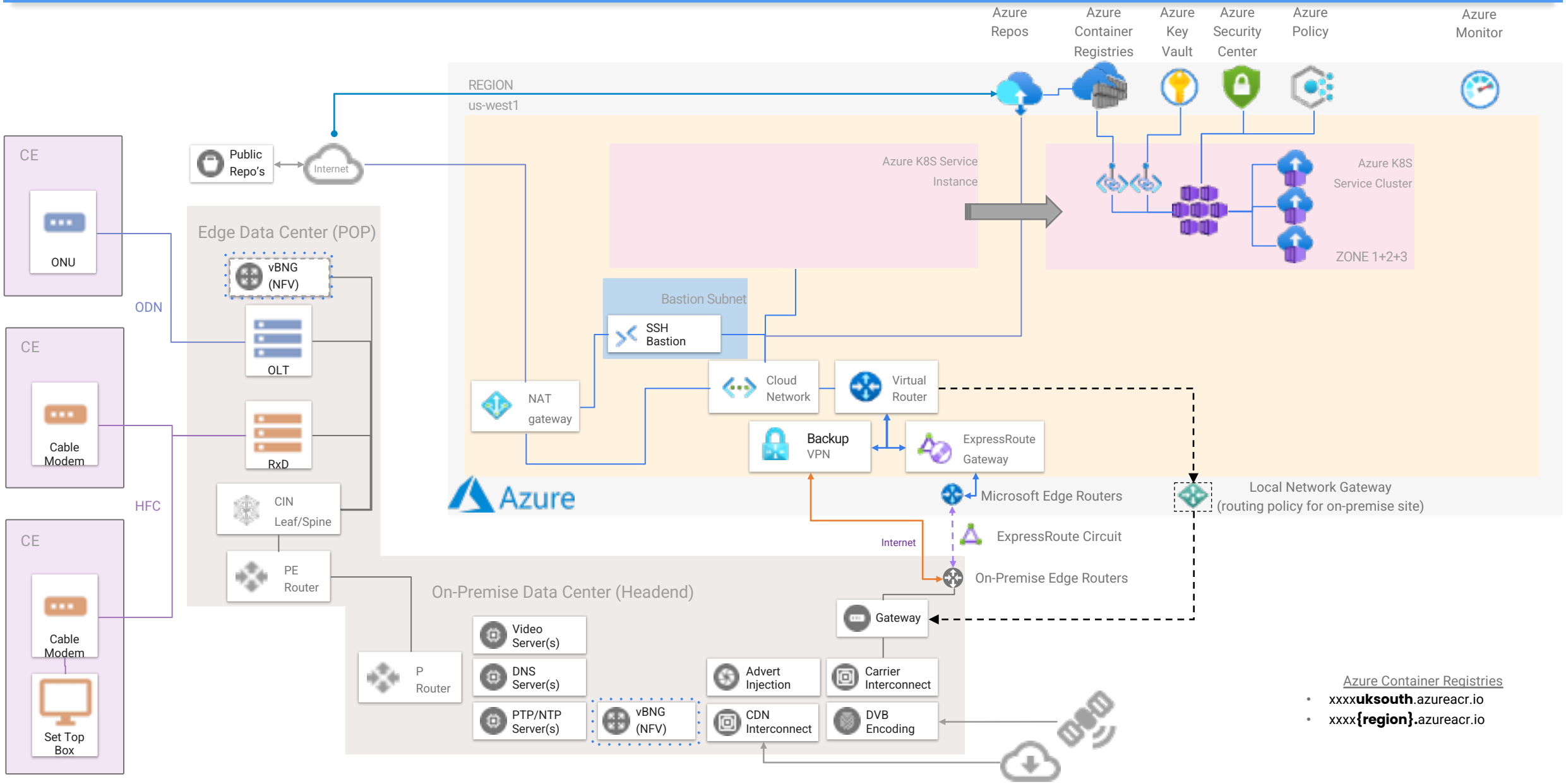
```
gcloud compute --project={PROJECT-ID} firewall-rules create bastion-ssh --direction=INGRESS --priority=1000 --network={NETWORK-NAME} --action=ALLOW --rules=tcp:22 --source-ranges={CS TECH SUPP IP}/32 --target-tags=bastion
```

- Create firewall rule to permit Bastion to interact with GKE

```
gcloud compute --project={PROJECT-ID} firewall-rules create bastion-fwd --direction=INGRESS --priority=1000 --network={NETWORK-NAME} --action=ALLOW --rules=all --source-tags=bastion
```

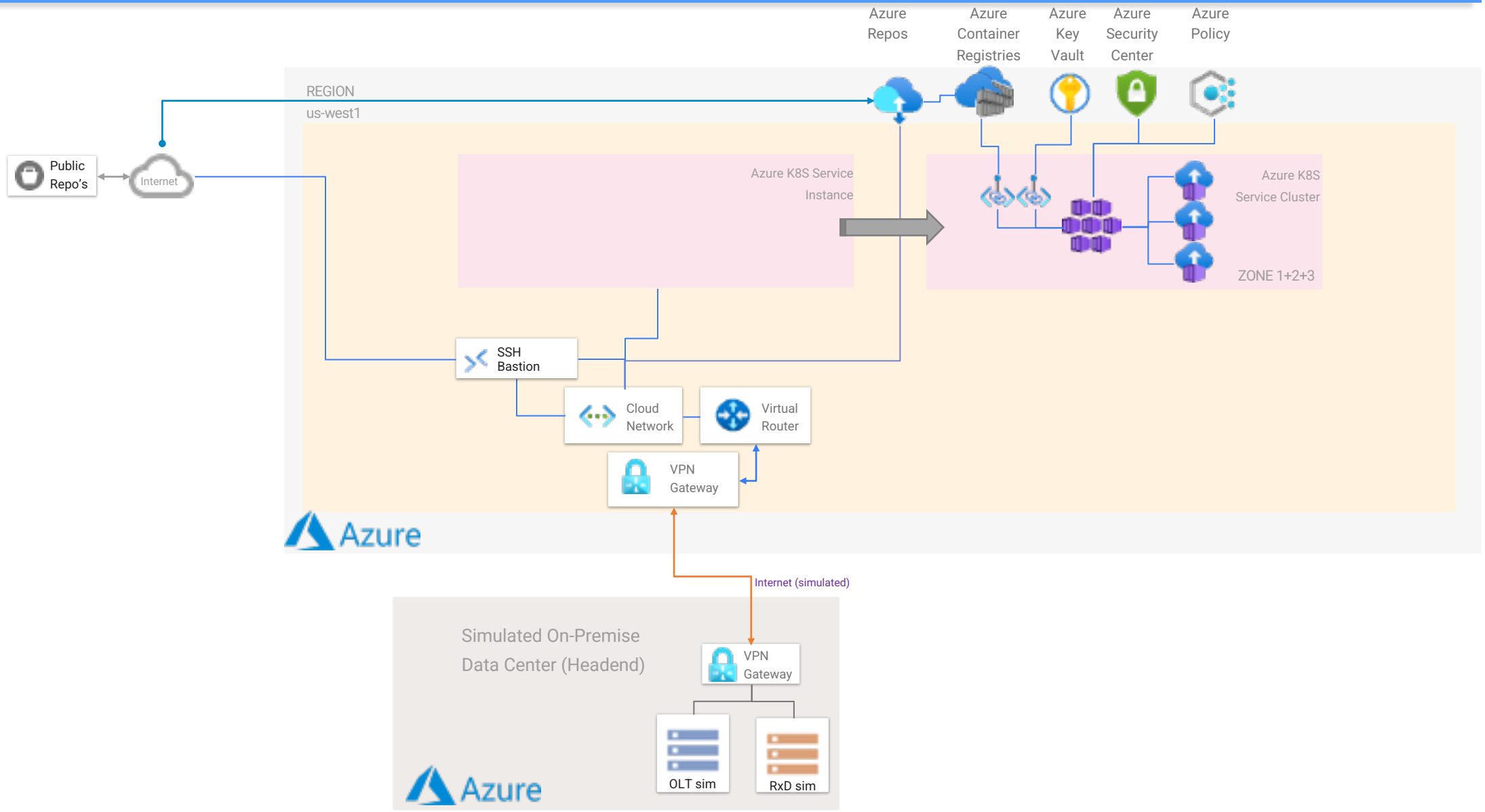
[20] Deploy

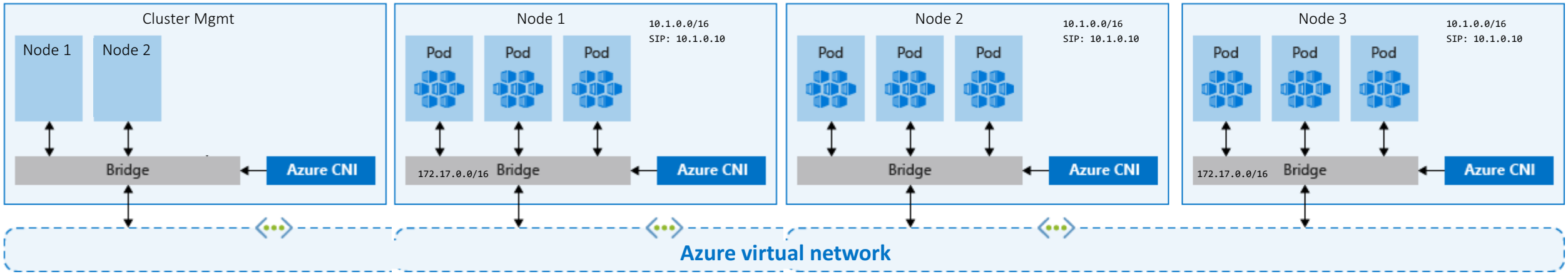
- Continue with existing deployment script for kubectl steps



Azure Container Registries

- xxxx**uksouth**.azureacr.io
- xxxx**{region}**.azureacr.io


Azure Repos
Azure Container Registries
Azure Key Vault
Azure Security Center
Azure Policy
REGION
us-west1
Public Repo's
Internet
Azure K8S Service Instance
Azure K8S Service Cluster
ZONE 1+2+3
SSH Bastion
Cloud Network
Virtual Router
VPN Gateway
Azure
Internet (simulated)
Simulated On-Premise Data Center (Headend)
VPN Gateway
OLT sim
RxD sim
Azure


Cluster Mgmt
Node 1
Node 2
Bridge
Azure CNI
Node 1
10.1.0.0/16
SIP: 10.1.0.10
Pod
Pod
Pod
172.17.0.0/16
Bridge
Azure CNI
Node 2
10.1.0.0/16
SIP: 10.1.0.10
Pod
Pod
Pod
Bridge
Azure CNI
Node 3
10.1.0.0/16
SIP: 10.1.0.10
Pod
Pod
Pod
172.17.0.0/16
Bridge
Azure CNI
Azure virtual network