Step 1: Verify the API

Step 2: Login to cloudshell

Step 3: Create a subnet in default VPC

```
gcloud compute networks subnets create gke-deep-dive-subnet --network=default --range=10.10.0.0/24
```

Step 4: Verify the created network and it’s subnet

No Secondary IP ranges

Step 5: Create VPC native cluster

```
gcloud container clusters create gke-deep-dive --num-nodes=1 --disk-type=pd-standard
--disk-size=10 --enable-ip-alias --subnetwork=gke-deep-dive-subnet --addons=HttpLoadBalancing
```

Step 6: Verify the secondary IP ranges created

Step 7: Verify the cluster is deployed okay

Step 8: In the console, check the VPC-native traffic routing under networking section of the GKE cluster - it should be enabled

The new cluster should have the `HttpLoadBalancing` add-on enabled, Check

If not, enable it by running the command below:

```
gcloud container clusters update gke-deep-dive --update-addons=HttpLoadBalancing=ENABLED
```

Step 9: Create necessary files:

1. **gke-deep-dive-app.yaml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gke-deep-dive-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: online
  template:
    metadata:
      labels:
        app: online
    spec:
      containers:
      - image: gcr.io/google-containers/echoserver:1.10
        name: gke-deep-dive-app
        ports:
          - name: http
            containerPort: 8080
        readinessProbe:
          httpGet:
            path: /healthz
            port: 8080
            scheme: HTTP
```

2. gke-deep-dive`-svc.yaml`:

```
apiVersion: v1

kind: Service

metadata:
  name: gke-deep-dive-svc
  annotations:
    cloud.google.com/l4-rbs: "enabled"
spec:
  type: LoadBalancer
  externalTrafficPolicy: Cluster
  selector:
    app: online
  ports:
  - name: tcp-port
    protocol: TCP
    port: 8080
    targetPort: 8080
```

Please notice the option: cloud.google.com/l4-rbs: "enabled"

It instructs GKE to create a backend service-based external passthrough Network Load Balancer **so that clients outside the cluster can send packets to the Service's Pods.**

Step 10: Create the deployment:

```
kubectl apply -f gke-deep-dive-app.yaml
```

Step 11: Verify that there are two serving Pods for the Deployment:

```
kubectl get pods
```

Step 12: Create the service:

```
kubectl apply -f gke-deep-dive-svc.yaml
```

Step 13: Verify that your Service is running:

```
kubectl get svc gke-deep-dive-svc
```

GKE assigned an `EXTERNAL_IP` for the external passthrough Network Load Balancer.

Note: It might take a few minutes for GKE to allocate an external IP address before the load balancer is ready to serve your application.

Test connecting to the load balancer:

```
curl EXTERNAL_IP:PORT
```

Step 14: Verify the external LoadBalancer Service and its components

```
    kubectl describe svc gke-deep-dive-svc
```

Step 15: Delete the `gke-demo-svc` external LoadBalancer Service.

```
kubectl delete service store-v1-lb-svc
```

Step 16: Delete the cluster

```
Gcloud container clusters delete gke-deep-dive
```