Kubernetes:

kubernetes-network-policy-recipes
https://github.com/ahmetb/kubernetes-network-policy-recipes
https://opensource.com/article/22/6/kubernetes-networking-fundamentals

https://blogs.vmware.com/vexpert/2021/12/19/12-days-of-kubernetes/

kubernetes-horizontal-pod-autoscaler-and-prometheus
https://www.weave.works/blog/kubernetes-horizontal-pod-autoscaler-and-prometheus

===================

Tutorial to Watch:
701411220#

1. GitOps Hands on to Build and Push Python App Docker Image to AWS ECR using GitHub Actions

https://www.youtube.com/watch?v=8RvQM0IZdoQ

Prometheus
https://www.youtube.com/watch?v=mLPg49b33sA&list=PLy7NrYWoggjxCF3av5JKwyG7FFF9eLeL4&index=3
9818566116
https://devopscube.com/setup-prometheus-monitoring-on-kubernetes/
https://gitlab.com/nanuchi/youtube-tutorial-series/-/blob/master/prometheus-exporter/commands.md

AWS in Hindi:
https://www.youtube.com/playlist?list=PLBGx66SQNZ8a_y_CMLHchyHz_R6-6i-i_

Comprehensive Nginx deployment to Kubernetes on AWS by using kops and terraform
https://www.udemy.com/course/learn-devops-kubernetes-deployment-by-kops-and-terraform/learn/lecture/18287718?start=0#overview

Kubernetes Certified Administrator (CKA) | GKE Deep Dive | BootStrap K8S & Prometheus with Istio | DevOps | HELM
https://www.udemy.com/course/kubernetes-training/learn/lecture/17457278?start=0#overview

How to deploy containers on Amazon ECS
https://www.udemy.com/course/learn-to-deploy-containers-on-aws-in-2021/learn/lecture/24358784?start=0#overview

Python
https://www.linkedin.com/learning/python-object-oriented-programming/basic-class-definition?autoAdvance=true&autoSkip=false&autoplay=true&resume=true&u=85418506

https://www.linkedin.com/learning/python-data-structures-dictionaries/python-dictionaries-and-why-you-should-use-them?autoAdvance=true&autoSkip=false&autoplay=true&resume=true&u=85418506

CKA
https://kodekloud.com/the-ultimate-guide-to-certified-kubernetes-administrator-cka-exam/

AWS
https://www.jerrychang.ca/writing/introducing-aws-ecs-technical-series?utm_source=linkedin

https://www.jerrychang.ca/writing/introducing-aws-ecs-technical-series

===================================
KUBERNETES: https://devopstitan.com/blog/

1. Kubernetes Rolling Deployment and rolling update
   https://devopstitan.com/cka-practice-question-10/?fbclid=IwAR0_-tkRRpv_477aXO9uuugz7_oep_or--rrBsjOv4--U8Y0YraQ6dxNoyM
2. How to Configure Network Policy
   https://devopstitan.com/cka-practice-question-08/

3. Services
   https://medium.com/swlh/kubernetes-services-simply-visually-explained-2d84e58d70e5

step -1: User executes a kubectl command.
step -2: kube-api server creates an unassigned pod object.
step -3: kube-api updates this info into the etcd.
step -4: After this update, the same is shared with the client.
step -5: kube-scheduler monitors and identifies an unassigned object
step -6: Scheduler identifies the right node to spin this object on
step -7: Communicates back to api-server with assigned node info.
step -8: API-server updates this new info into etcd.
step -9: Api-server proceeds to assign object to respective kubelet
step -10: Kubelet creates pod(s) and spins container in them via CRI

step -11: kubelet updates api-server. This info is updated in etcd.
step -12: Controller-Manager uses api-server to monitor k8s objects.

Interview Questions:
https://kodekloud.com/blog/top-kubernetes-interview-questions/

https://github.com/cloudnloud/interview-questions/blob/main/kubernetes/k8s-interview-questions-part-1.md

https://github.com/cloudnloud/interview-questions/blob/main/terraform/terraform-interview-questions-part-1.md

Python
https://www.interviewbit.com/python-interview-questions/

S3
Istio is an open-source service mesh-a modernized service networking layer that provides a transparent way to easily and flexibly automate application network processes. It layers transparently onto existing distributed applications. Istio's core concepts are Traffic Management, Observability, and Security capabilities.

Why use Istio?

Istio helps organizations run distributed, microservices-based apps anywhere. Istio's powerful features provide a unified and more efficient way to secure, connect, and monitor Kubernetes services.

- Service Mesh is the cloud-native counterpart of TCP/IP. It facilitates application network communication, visibility, and security.
- Istio is the most popular service mesh implementation out there at the moment, relying on Kubernetes but also scalable to virtual machine loads.
- Istio acts as the network layer of the cloud infrastructure and is transparent to applications.

What is the Istio service mesh used for?

Istio manages traffic flows between services, aggregates data, and reinforces access policies, with few to no changes to the application code. It mitigates deployment complexity as it layers onto existing distributed applications transparently.
Istio empowers load balancing, service-to-service authentication, and

monitoring. Istio's powerful control plane brings crucial features, such as:

- Secure service-to-service communication in a cluster with TLS encryption, strong identity-based authorization, and authentication.
- Automatic load balancing.
- Finely segregated control of traffic behavior with rich routing rules, fault injection, and failovers.
- A policy layer and configuration API that support access control, rate limits, and quotas.
- Automatic logs, metrics, and traces for the whole traffic within a cluster, including cluster ingress and egress.

What is the use of Istio in Kubernetes?

Kubernetes is essentially about managing the application lifecycle through declarative configuration, while a service mesh provides inter-application traffic, and improves security management and observability. Once you dispose of an application platform using Kubernetes, the Istio service mesh eases the implementation of load balancing and traffic control for calls between services. Istio adjuncts Kubernetes, by increasing its traffic management, security, and observability for cloud-native distributed applications.

Istio

Istio is a Kubernetes-native service mesh initially developed by Lyft and highly adopted in the industry. Leading cloud Kubernetes providers like Google, IBM, and Microsoft use Istio as the default service mesh in their services. Istio provides a robust set of features to create connectivity between services, including request routing, timeouts, circuit breaking, and fault injection. Additionally, Istio creates deep insights into applications with metrics such as latency, traffic, and errors.

Pros	**Cons**
The most active community High adoption in the industry Works with Kubernetes and VMs	Steep learning curve Significant overhead to the cluster No native admin dashboard

Servicemesh

Service mesh technology emerged with the popularization of microservice architectures. Because service mesh facilitates the separation of networking from the business logic, it enables you to focus on your application's core competency.
Microservice applications are distributed over multiple servers, data centers, or continents, making them highly network dependent. Service mesh manages network traffic between services by controlling traffic with routing rules and the dynamic direction of packages between services.
=============
Troubleshooting Kubernetes Networking – Part1

Tweet on Twitter

FOXUTECH
TROUBLESHOOTING KUBERNETES NETWORKING
Kubernetes Troubleshooting Series
Part 1

Troubleshooting Kubernetes Networking: As we are seeing one by one issues on Kubernetes and how to fix it, part of that today we are going to see, another important topic, which is nothing but networking. As this is vest topic to cover, we will be separate one by one and provide the solutions. In this topic, we will be taking some example scenarios and how to fix it.
In this series we should get understand the basics and few related topics, then will see very basic issue and how to fix in this, in coming post will be seeing another extended issue and how to fix it.

Basics

Services

Before we proceed, let's check about Kubernetes service, A Kubernetes service is a logical abstraction for a deployed group of pods in a cluster (which all perform the same function). Since pods are ephemeral, a service enables a group of pods, which provide specific functions (web services, image processing, etc.) to be assigned a name and unique IP address (clusterIP). As long as the service is running that IP address, it will not change. Services also define policies for their access.

Types of Kubernetes services?

- **ClusterIP**. Exposes a service which is only accessible from within the cluster.
- **NodePort**. Exposes a service via a static port on each node's IP.
- **LoadBalancer**. Exposes the service via the cloud provider's load balancer.
- **ExternalName**. Maps a service to a predefined externalName field by returning a value for the CNAME record.

Container Network Interface

As we know, Every Pod in a cluster gets its own unique cluster-wide IP address. This means you do not need to explicitly create links between Pods and you almost never need to deal with mapping container ports to host ports. This creates a clean, backwards-compatible model where Pods can be treated much like VMs or physical hosts from the perspectives of port allocation, naming, service discovery, load balancing, application configuration, and migration. Kubernetes imposes the following fundamental requirements on any networking implementation (barring any intentional network segmentation policies): pods can communicate with all other pods on any other node without NAT agents on a node (e.g. system daemons, kubelet) can communicate with all pods on that node.

Kubernetes networking addresses four concerns:

- Containers within a Pod use networking to communicate via loopback.
- Cluster networking provides communication between different Pods.
- The Service resource lets you expose an application running in Pods to be reachable from outside your cluster.
- You can also use Services to publish services only for consumption inside your cluster.

CNI Plugins:

Here are some familiar or most used CNI plugins,

1. Flannel
2. Calico
3. Cilium
4. WeaveNet
5. Canal

Summary Matrix

	Flannel	Calico	Cilium	Weavenet	Canal
Mode of Deployment	DaemonSet	DaemonSet	DaemonSet	DaemonSet	DaemonSet
Encapsulation and Routing	VxLAN	IPinIP,BGP,eBPF	VxLAN,eBPF	VxLAN	VxLAN
Support for Network Policies	No	Yes	Yes	Yes	Yes
Datastore used	Etcd	Etcd	Etcd	No	Etcd
Encryption	Yes	Yes	Yes	Yes	No

Ingress Support	No	Yes	Yes	Yes	Yes
Enterprise Support	No	Yes	No	Yes	No

How to choose a CNI Provider?

There is no single CNI provider that meets all our project needs, here some details about each provider. For easy setup and configuration, Flannel and Weavenet provide great capabilities. Calico is better for performance since it uses an underlay network through BGP. Cilium utilizes a completely different application-layer filtering model through BPF and is more geared towards enterprise security.

Basic Troubleshooting

Traffic

As we have seen, Kubernetes supports a variety of networking plugins as seen above and each fails in its own way. To troubleshoot the issues in networking we should understand the core. Kubernetes relies on the Netfilter kernel module to set up low level cluster IP load balancing. This requires two critical modules, IP forwarding and bridging.

Kernel IP forwarding

IP forwarding is a kernel setting that allows forwarding of the traffic coming from one interface to be routed to another interface. This setting is necessary for Linux kernel to route traffic from containers to the outside world.

What it causes?

Sometimes this setting could be reset by a security team running while security scans/enforcements or some system changes, or have not been configured to survive a reboot. When this happens networking starts failing.

Pod to service connection times out:

```
* connect to 10.0.21.231 port 3000 failed: Connection timed out
* Failed to connect to 10.0.21.231 port 3000: Connection timed out
* Closing connection 0
curl: (7) Failed to connect to 10.0.21.231 port 3000: Connection timed out
```

Tcpdump could show that lots of repeated SYN packets are sent, but no ACK is received.

How to diagnose

Check that ipv4 forwarding is enabled

```
# sysctl net.ipv4.ip_forward
```

0 means that forwarding is disabled

```
net.ipv4.ip_forward = 0
```

How to fix

This will turn things back on a live server

```
# sysctl -w net.ipv4.ip_forward=1
```

on Centos/RHEL this will make the setting apply after reboot

```
# echo net.ipv4.ip_forward=1 >> /etc/sysconf.d/10-ipv4-forwarding-on.conf
```

Bridge-netfilter

The bridge-netfilter setting enables iptables rules to work on Linux bridges just like the ones set up by Docker and Kubernetes. This setting is necessary for the

Linux kernel to be able to perform address translation in packets going to and from hosted containers.

What it causes?

Network requests to services outside the Pod network will start timing out with destination host unreachable or connection refused errors.

How to diagnose

Check that bridge netfilter is enabled

```
sysctl net.bridge.bridge-nf-call-iptables
```

0 means that bridging is disabled

```
net.bridge.bridge-nf-call-iptables = 0
```

How to fix

Note some distributions may have this compiled with kernel, check with

```
# cat /lib/modules/$(uname -r)/modules.builtin | grep netfilter
# modprobe br_netfilter
```

Turn the iptables setting on

```
# sysctl -w net.bridge.bridge-nf-call-iptables=1
# echo net.bridge.bridge-nf-call-iptables=1 >> /etc/sysconf.d/10-bridge-nf-call-iptables.conf
```

Firewall rules block overlay network traffic

Kubernetes provides a variety of networking plugins that enable its clustering features while providing backwards compatible support for traditional IP and port based applications.

One of most common on-premises Kubernetes networking setups leverages a VxLAN overlay network, where IP packets are encapsulated in UDP and sent over port 8472.

What it causes?

There is 100% packet loss between pod IPs either with lost packets or destination host unreachable.

```
$ ping 10.22.192.108
PING 10.22.192.108 (10.22.192.108): 56 data bytes
--- 10.22.192.108 ping statistics ---
5 packets transmitted, 0 packets received, 100% packet loss
```

How to diagnose

It is better to use the same protocol to transfer the data, as firewall rules can be protocol specific, e.g. could be blocking UDP traffic.

iperf could be a good tool for that:

on the server side

```
# iperf -s -p 5432 -u
```

on the client side

```
# iperf -c 10.22.192.108 -u -p 5432 -b 1K
```

How to fix

Update the firewall rule to stop blocking the traffic. Here is some common iptables advice.

Pod CIDR conflicts

Kubernetes sets up special overlay network for container-to-container communication. With isolated pod network, containers can get unique IPs and avoid port conflicts on a cluster. The problems arise when Pod network subnets start conflicting with host networks.

What it causes?

Pod to pod communication is disrupted with routing problems.

```
# curl http://172.24.28.32:3000
curl: (7) Failed to connect to 172.24.28.32 port 3000: No route to host
```

How to diagnose

Start with a quick look at the allocated pod IP addresses:

```
# kubectl get pods -o wide
```

Compare host IP range with the kubernetes subnets specified in the apiserver:

```
# ip addr list
```

IP address range could be specified in your CNI plugin or kubenet pod-cidr parameter.

How to fix

Double-check what RFC1918 private network subnets are in use in your network, VLAN or VPC and make certain that there is no overlap.

Once you detect the overlap, update the Pod CIDR to use a range that avoids the conflict.