



PODMAN 是一个容器引擎

- 支持多种操作系统
- 支持多种硬件架构
- 支持多种网络配置
- 支持多种存储配置



POD?

Pod 是什么

Pod 是 Kubernetes 的最小调度单元。一个 pod 包含一个或多个 container (app)。一个 pod 中的所有 container 共享相同的网络 namespace。

CRI

Container Runtime Interface

metric

指标

secret

密钥。用于存储敏感信息。

Persistent Volume (PV)

持久化存储。用于存储需要长期保存的数据。与 Pod 的生命周期无关。

PV 由集群管理员创建。Pod 可以请求使用 PV。

YAML

配置文件格式。用于定义 Kubernetes 资源。podman (类似 docker) 不使用 Kubernetes。

YAML 是纯文本格式，tab 和 space 都可以。tab 只能有一个 identity provider can

YAML 在 vim 中设置 :set cursorcolumn 可以显示行号。

```
:set cursorcolumn
```

## Orchestration

容器编排工具。用于管理容器的生命周期，包括启动、停止、重启、扩缩容等。

容器编排工具包括 Kubernetes, OpenShift, Rancher, Kubesphere 等。其中 Kubernetes 是最流行的容器编排工具。

## ingress

用于管理容器网络流量的工具。通常用于将外部流量路由到容器集群中的特定服务。

常见的 Ingress 控制器包括 Nginx Ingress Controller, Traefik, Istio 等。

## egress

用于管理容器网络流量的工具。通常用于将容器集群中的流量路由到外部网络。

## etcd

etcd 是一个分布式键值数据库，用于存储 Kubernetes 集群的元数据。

etcd 是 Kubernetes 集群的基石，用于存储集群的元数据，包括节点信息、Pod 信息、Service 信息等。

CNCF 基金会孵化了 Kubernetes 项目。Kubernetes 是一个开源的容器编排系统。COREOS 公司开发了 Kubernetes 的早期版本。

容器编排

容器编排

## scale-up 和 scale-out

out: 增加容器的数量，通常用于处理突发流量。

up: 增加容器的 CPU 和内存资源，通常用于处理持续高负载。

## statefull vs stateless

statefull vs stateless vs stateless

statefull - stateful

stateless - stateless

stateless vs stateful APP vs stateful APP vs stateful APP.

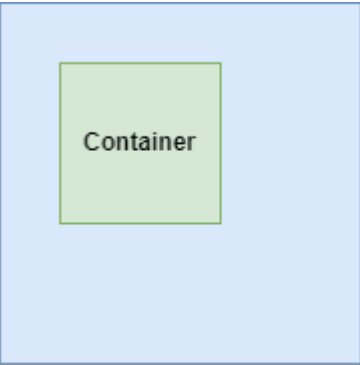
log stateful, stateful DB vs stateful APP vs stateful.

SAN stateful vs ReadWriteOnce vs stateful stateful.



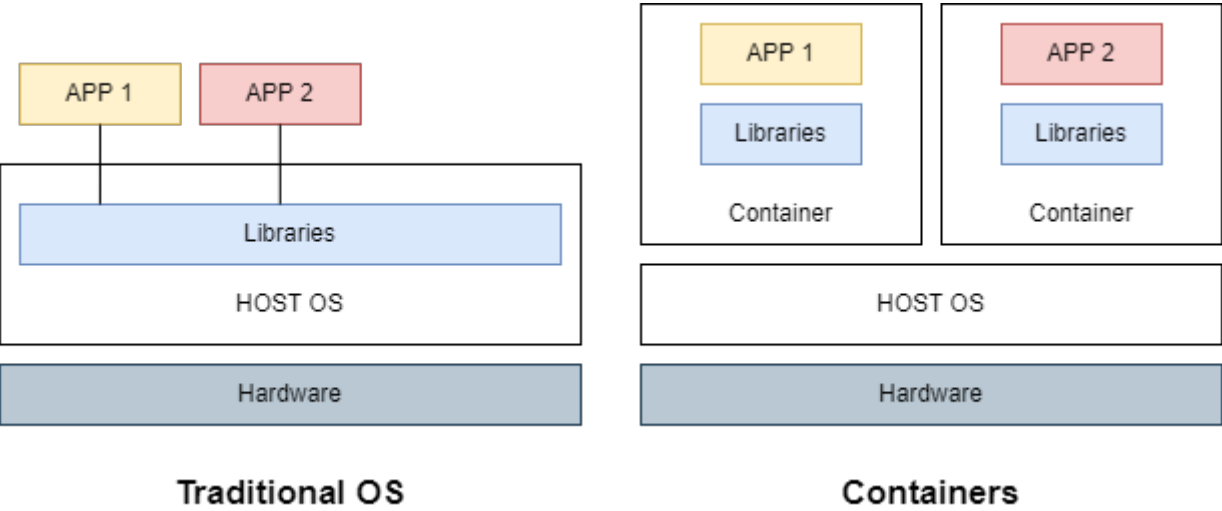
POD 是 一个 容器 的 集合。

每个 pod 包含 container (app) 以及 它们 共享 的 资源，每个 pod 可以 包含 多个 container。



POD

# APP 如何 在 OS 中 运行



在 传统 OS 中，APP 1 和 APP 2 共享 相同的 库。而在 容器 中，APP 1 和 APP 2 各自拥有 自己的 库。

每个 APP 都 拥有 自己 的 库，这 使得 OS 可以 更好地 管理 资源。

이 글에서는 애플리케이션(APP) 컨테이너(POD)가 어떻게 운영체제(OS)를 가상화하는지 설명합니다.

## Virtualization과 containerization의 차이

가상화란 하드웨어를 소프트웨어로 모방하는 것을 의미하며, VM(가상머신)은 운영체제를 가상으로 실행합니다. 컨테이너는 운영체제 위에서 실행되는 애플리케이션을 격리하는 기술입니다.

## Rootful 컨테이너와 rootless 컨테이너의 차이

Rootful 컨테이너는 호스트 OS의 root 권한을 가진 컨테이너이며, Rootless 컨테이너는 호스트 OS의 root 권한 없이 실행되는 컨테이너입니다.

이 글에서는 Rootful 컨테이너와 Rootless 컨테이너의 차이점을 설명하고, 각각의 장단점을 비교합니다.

- POD는 Rootful 컨테이너와 Rootless 컨테이너로 나뉘며, Rootful 컨테이너는 호스트 OS의 root 권한을 가진 컨테이너입니다.
- HOST OS는 1~1023개의 프로세스를 실행할 수 있으며, Rootful 컨테이너는 이 프로세스를 실행할 수 있습니다.
- POD는 Rootful 컨테이너를 실행할 수 있습니다.

# PODMAN

-d

detach 背景で実行される。

-d は POD を foreground で実行するのではなく、背景で実行する。背景で実行する場合は、-d は POD を background で実行する。背景で実行する場合は、-d は POD を background で実行する。

-p 80:8080

80 : HOST OS のポート

8080 : コンテナのポート

-v :Z

-v :Z は SELinux のラベルを保持する。

podman

podman は podman のコマンドを実行する。

podman --log-level=debug info

podman save

local に保存する tar ファイル

podman load

tar ファイルを local にロードする。tar ファイルは local に保存されている。

podman exec 容器 容器

podman exec 容器 容器 容器 容器 容器.

/bin/bash 容器 容器 容器 容器 容器. (podman exec 容器 Ghost 容器)

podman **exec** -it <POD 容器 ID> /bin/bash

容器) mariadb 容器 POD 容器

podman exec -it mariadb /bin/bash

mysql -u root -p





### 反向代理

反向代理是指将客户端的请求转发给后端服务器。通常使用 apache、nginx 等软件实现，nginx 是常用的 web 代理，它支持反向代理 reverse proxy 功能，可以将 HOST port 转发给后端服务器。

例：1 个 POD 8080, 2 个 POD 8081

# 1. Docker 部署

## docker 部署步骤

docker 部署步骤如下 4 个步骤，每个步骤都有对应的命令。

- 1. EXPOSE -p 指定端口
- 2. EXPOSE 指定端口
- 3. EXPOSE -p 指定端口
- 4. -p 指定端口

- 步骤 1**  
在 docker 中运行容器时，需要指定 POD 的端口。通常使用 docker run 命令，指定 POD 的端口，例如：docker run -p 8080:8080 nginx。其中 8080 是宿主机的端口，8080 是容器的端口。
- 步骤 2 --expose**  
在 docker 中运行容器时，可以使用 --expose 选项来指定容器的端口。例如：docker run --expose 8080 nginx。这样容器就会暴露 8080 端口。
- 步骤 3 --publish-all**  
在 docker 中运行容器时，可以使用 --publish-all 选项来指定容器的端口。例如：docker run --publish-all nginx。这样容器的所有端口都会暴露给宿主机的所有端口。
- 步骤 4 --publish (-p)**  
在 docker 中运行容器时，可以使用 --publish (-p) 选项来指定容器的端口。例如：docker run -p 8080:8080 nginx。这样容器的 8080 端口就会暴露给宿主机的 8080 端口。

## 2. PODMAN 简介

podman 是 docker 的替代品，它暴露了 POD 的接口，但不需要 root 权限。

podman publish 命令用于发布容器镜像，rootful podman 需要 root 权限。

安装

podman network ls

port 命令

podman port -a

POD 简介

容器在 HOST OS 上运行，不需要 root 权限。

podman network reload

POD 简介

容器在 HOST OS 上运行，不需要 root 权限。

podman network inspect <网络名称>

podman network inspect podman