破解Kubernetes应用开发困局

实时热加载和一键 Debug

CLOUD NATIVE ACADEMY 2020

云原生学院第21期

王炜

腾讯云CODING DevOps高级架构师

CNCF大使

8月5日（周四）晚 8 点-9 点

扫码提问

主办方：云原生社区 Cloud Native Community　直播平台：bilibili　互动平台：腾讯文档

自我介绍

腾讯云 CODING DevOps 高级架构师

CNCF 大使

Nocalhost 项目负责人

目录

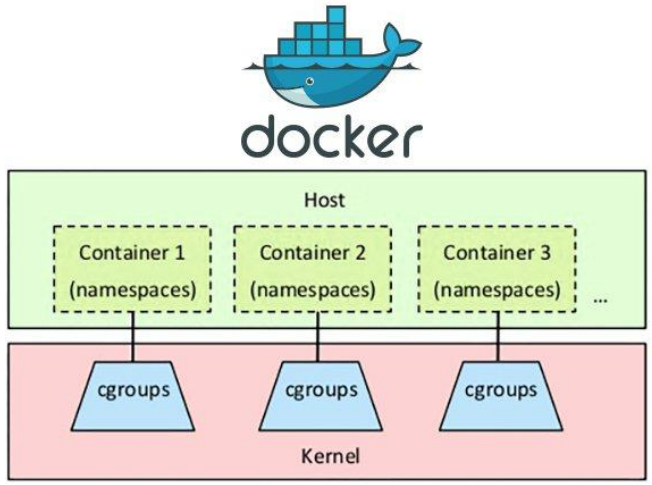
01

K8s 环境开发困局

开发举步维艰

K8s 环境的开发困局

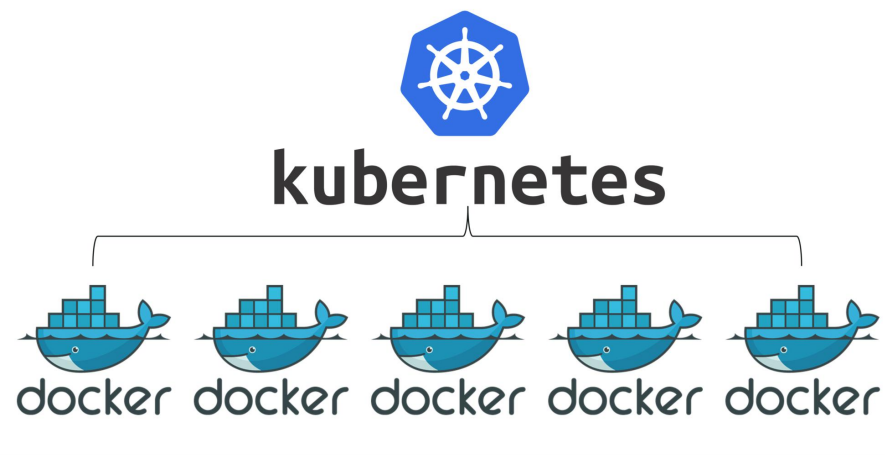
微服务–Docker



微服务越来越多，运行环境变复杂。服务依赖、打包、运行、迁移越来越难。

Docker 提供镜像打包的解决方案。

Docker–Kubernetes



容器越来越多，服务编排、发现、稳定性监控、自愈等成为新的挑战。

Kubernetes 提供容器编排的解决方案。

面向运维

- 开发难

概念繁多，声明式定义学习成本高。

- 调试难

无法像本地一样调试，开发效率低。

```
    image: codingcorp-docker.pkg.coding.net/registry/build/cci:{{ .Values.imageVersion.ci | default "master" }}
    livenessProbe:
      exec:
        command:
        - bash
        - -c
        - /bin/grpc_health_probe --addr=:15733
      failureThreshold: 30
      initialDelaySeconds: 60
      periodSeconds: 10
      successThreshold: 1
      timeoutSeconds: 3
    readinessProbe:
      exec:
        command:
        - bash
        - -c
        - /bin/grpc_health_probe --addr=:15733
      failureThreshold: 30
      initialDelaySeconds: 60
      periodSeconds: 10
      successThreshold: 1
      timeoutSeconds: 3
    resources:
      {{- if .Values.global.resourceRequest.enabled }}
      limits:
        cpu: 700m
        memory: 1536Mi
      requests:
        cpu: 400m
        memory: 512Mi
      {{- end }}
```

完全面向运维提供能力，对开发增加了巨大的负担。

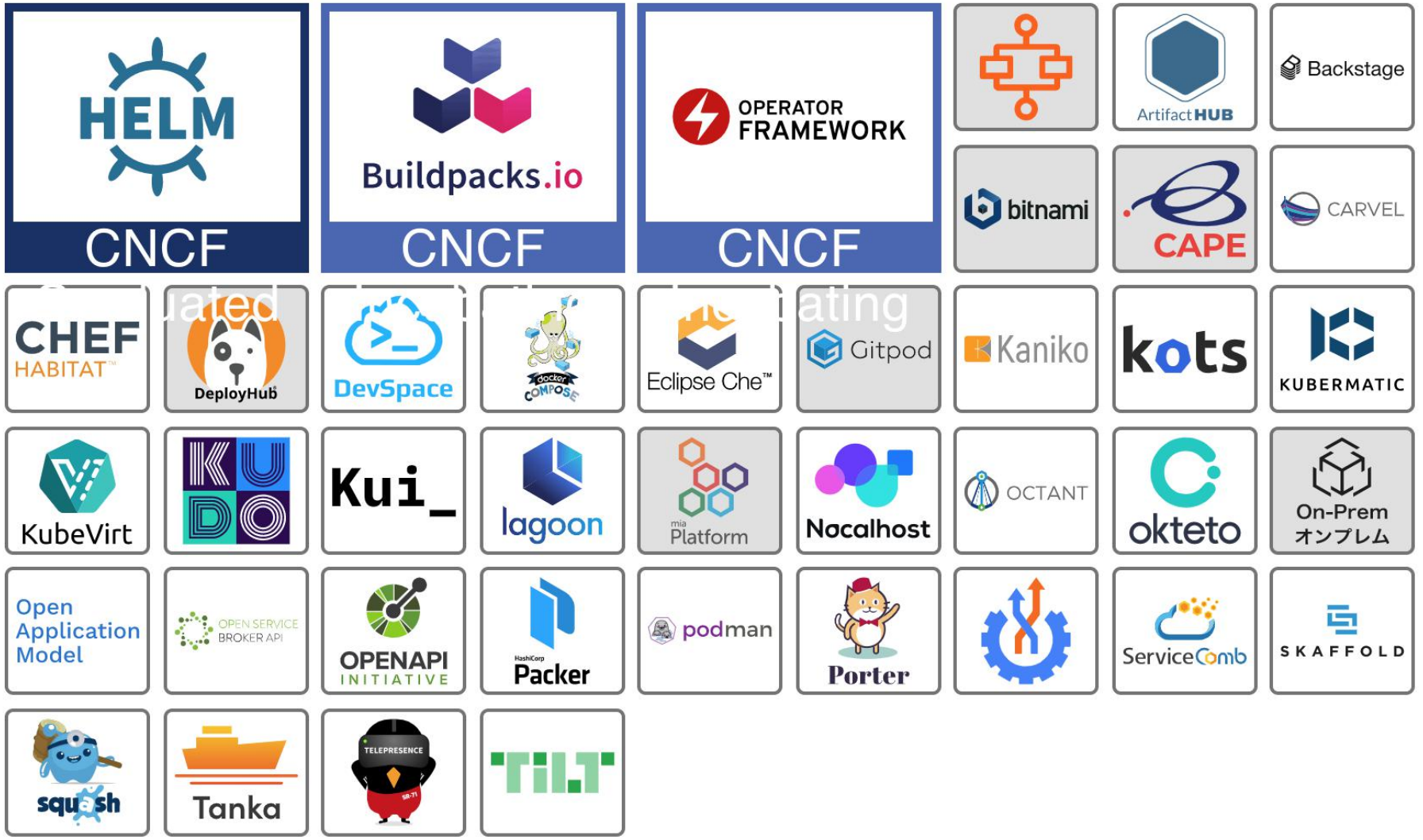
云原生环境下的学习成本，招聘成本，用人成本急剧上升。

云原生开发技能广度要求急剧提升

传统应用后端必备技能

云原生应用后端必备技能

云原生开发工具依然缺失

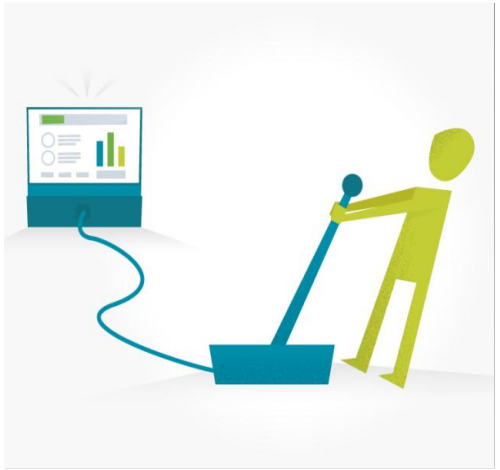


02

主流云原生开发方式

现状

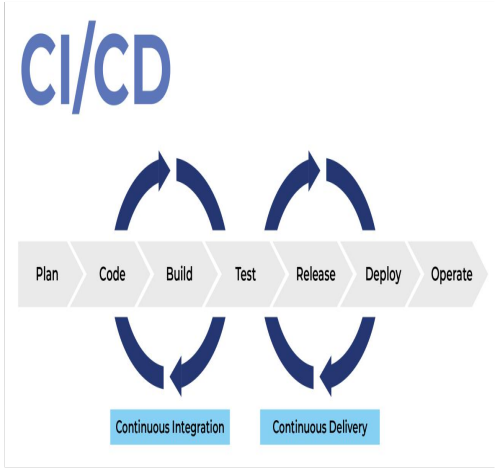
主流的云原生开发方式（开发环境）



全手工流程

编码后，手动构建镜像、推送到镜像仓库、修改工作负载镜像版本，调度

10 分钟/次



自动化 CI/CD 流程

编码后，推送到代码仓库，自动触发 CI/CD 流程，等待生效。

5 分钟/次

Minikube + Telepresence

Minikube 拉起本地 K8s 开发环境，Telepresence 实现本地编码。

10 秒/次

云环境 + Telepresence

云上 K8s 集群提供计算资源解决弹性的问题，Telepresence 本地编码。

10 秒/次

Telepresence 局限性

（官网推荐的开发方式）

本地环境和容器、工作负载声明有很大的差异，导致业务源码很难在本地运行。

环境差异

工作负载声明了 env、configmap、secret、volume 等，很难在本地复制出完全一致的环境。

跨平台差异

即便是能够将远端的 env、configmap 挂载到本地，也难以屏蔽跨平台之间的差异。

网络限制

全量代理的方式会使得网络拓扑产生变化，导致内网、公网访问无法达到预期。

03

热加载原理

实现容器内应用/进程热加载

从 Dockerfile 说起

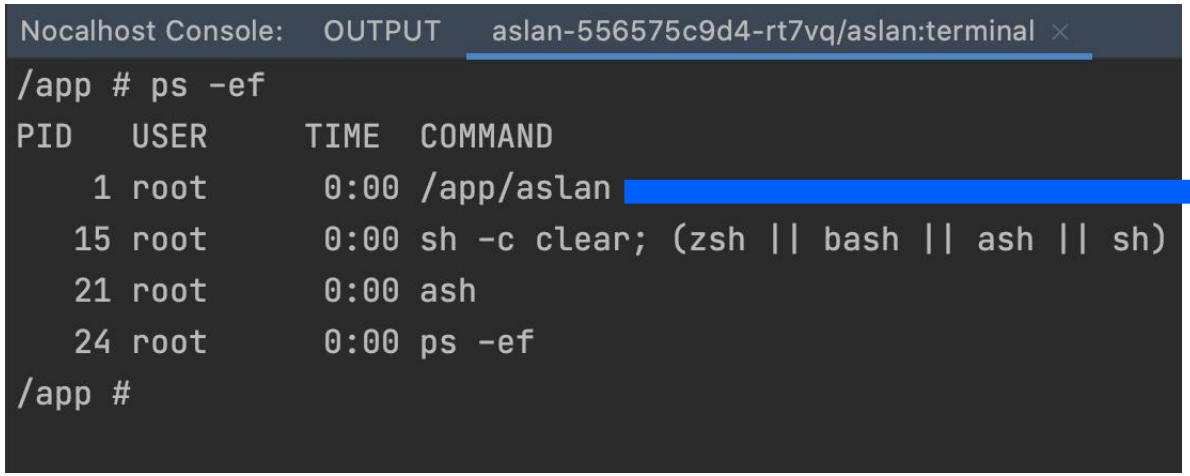
```
# https://wiki.alpinelinux.org/wiki/Setting_the_timezone
RUN sed -i 's/dl-cdn.alpinelinux.org/mirrors.aliyun.com/g'
    apk add tzdata && \
    cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime &&
    echo Asia/Shanghai  > /etc/timezone && \
    apk del tzdata

WORKDIR /app

COPY --from=build /aslan .

ENTRYPOINT ["/app/aslan"]
```

```
Nocalhost Console:   OUTPUT   aslan-556575c9d4-rt7vq/aslan:terminal
/app # ps -ef
PID   USER     TIME  COMMAND
    1 root      0:00 /app/aslan
   15 root      0:00 sh -c clear; (zsh || bash || ash || sh)
   21 root      0:00 ash
   24 root      0:00 ps -ef
/app #
```

Dockerfile CMD 或 ENTRYPOINT 定义容器启动命令

对应容器 PID=1 的进程

从 Dockerfile 说起



go run cmd/aslan/main.go

缺少：源码、Golang Runtime

从 Dockerfile 说起

还缺三个条件：

1、源码从哪来?

2、Golang Runtime 从哪来?

3、PID=1 的进程替换成源码运行，如果进程停止，容器将 Crash，怎么阻止?

解决问题：

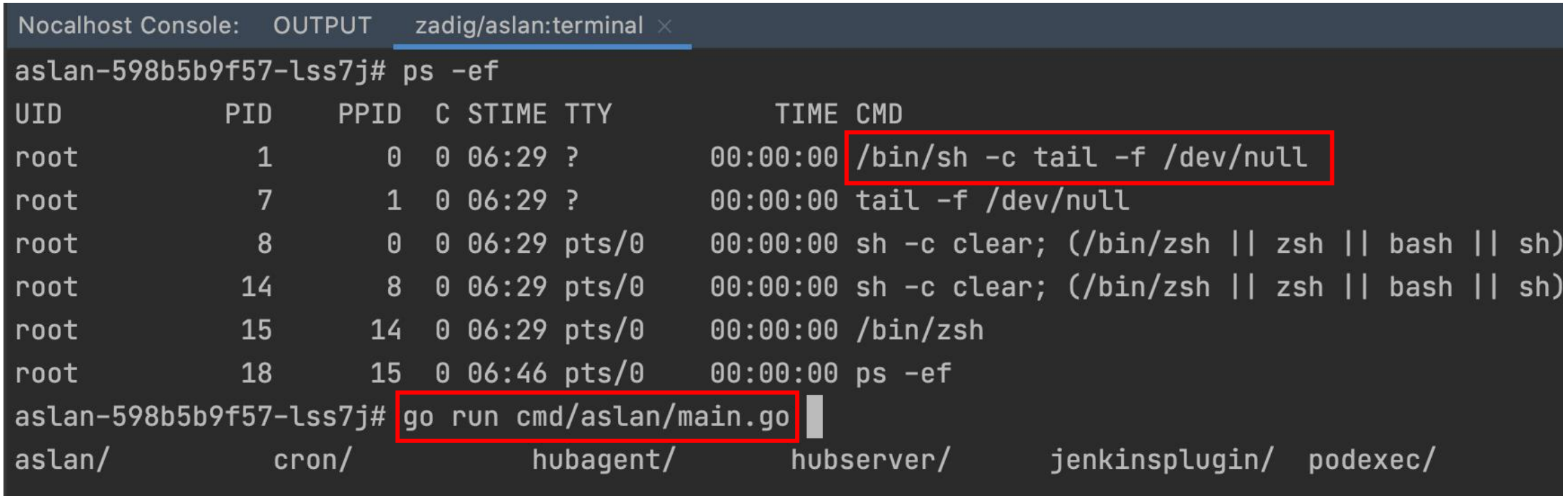
1、从本地同步到容器

2、将业务容器的镜像替换为 Runtime 镜像

3、替换 PID=1 进程为阻塞进程：

/bin/sh –c tail –f /dev/null

从 Dockerfile 说起



04

开发和调试演示

一键 Run、一键 Debug

使用 Nocalhost 开发 Zadig 组件

教程：https://koderover.com/tutorials/

Bilibili：

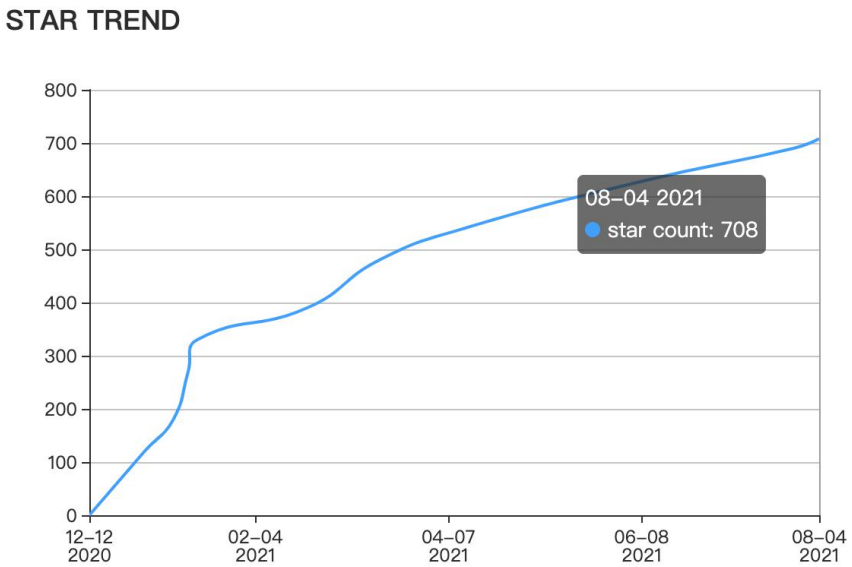
https://www.bilibili.com/video/BV1Sq4y1H7B1?from=search&seid=182889405009594269 55

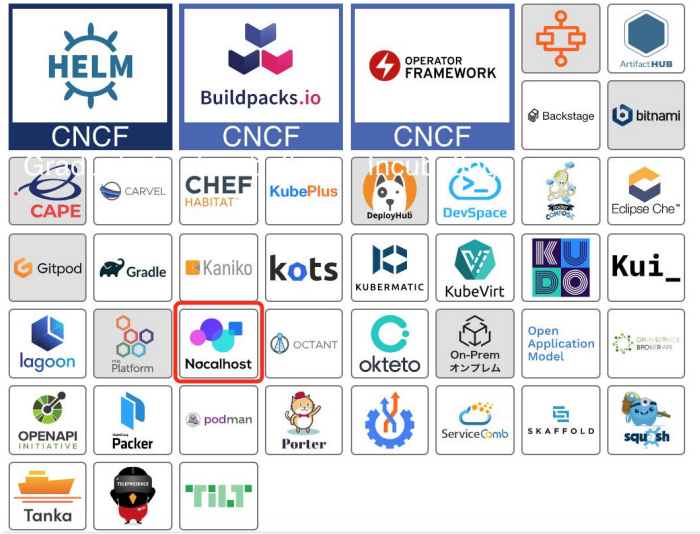
05

开源共建

100% 开源

开源共建


STAR TREND
800
700
600
500
400
300
200
100
0
12–12
2020
02–04
2021
04–07
2021
06–08
2021
08–04
2021
08–04 2021
star count: 708


HELM
CNCF
Buildpacks.io
CNCF
OPERATOR FRAMEWORK
CNCF
Backstage
bitnami
CAPE
CARVEL
CHEF
HABITAT
KubePlus
DeployHub
DevSpace
Eclipse Che
Gitpod
Gradle
Kaniko
kots
KUBERMATIC
KubeVirt
Kui_
lagoon
Nocalhost
OCTANT
okteto
On-Prem
Open
Application
Model
OPENAPI
INITIATIVE
Packer
podman
Porter
ServiceComb
SKAFFOLD
squash
Tanka
TILT