NDX

NONE

Ting Ye

DaoCloud

TABLE OF CONTENTS

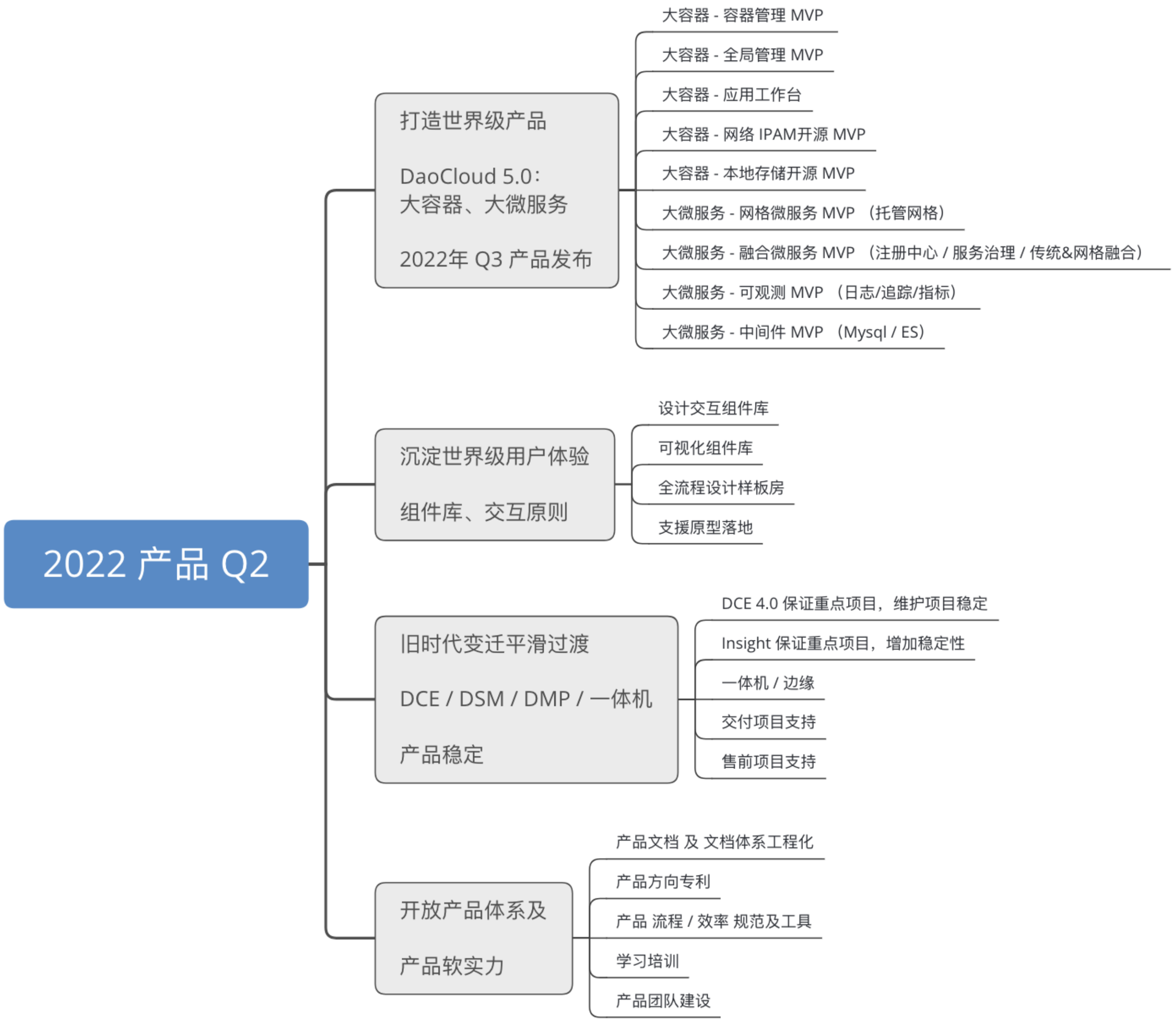
1. HOME

1.1 NDX

NDX DCE DSM DMP DaoCloud Enterprise 5.0

NDX / /

2022 Q2



: 2022-05-08

1.2 SUMMARY

- NDX

1.2.1

- DCE 5.0
-
-
- ClusterPedia
-
- HwameiStor
- UX
-
-
-
- Merbridge
-
-
-
- Kube Grid
-
-
-
-
-
-
- DCE 4.0
- Insight
- /
-
-
-
-
-
- /
-
-
-
- CCE
- KubeSphere
- Tanzu
- Rancher

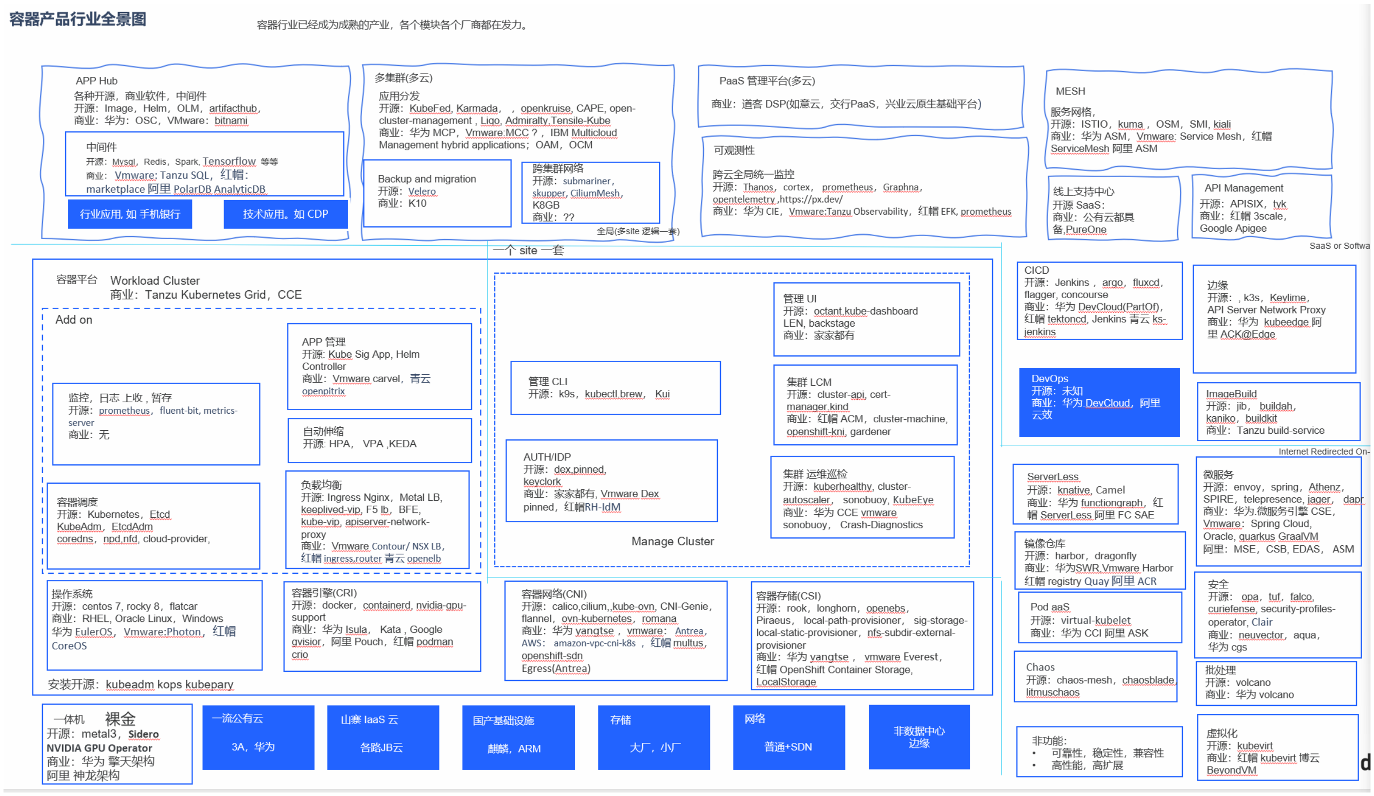
- Cilium
-
- Karmada

: 2022-05-08

2. DCE 5.0

2.1 DAOCLOUD ENTERPRISE 5.0

DaoCloud Enterprise 5.0   DCE 5.0   DaoCloud


容器产品行业全景图
容器行业已经成为成熟的产业，各个模块各个厂商都在发力。
APP Hub
各种开源，商业软件，中间件
开源：Image，Helm，OLM，artifacthub，
商业：华为：OSC，VMware：bitnami
中间件
开源：Mysql，Redis，Spark, Tensorflow 等等
商业：Vmware: Tanzu SQL，红帽：marketplace 阿里 PolarDB AnalyticDB
行业应用, 如 手机银行
技术应用。如 CDP
多集群(多云)
应用分发
开源：KubeFed, Karmada，，openkruise, CAPE, open-cluster-management，Ligo, Admiralty,Tensile-Kube
商业：华为 MCP，Vmware:MCC？，IBM Multicloud Management hybrid applications；OAM，OCM
Backup and migration
开源：Velero
商业：K10
跨集群网络
开源：submariner，skupper, CiliumMesh，K8GB
商业：??
全局(多site 逻辑一套)
PaaS 管理平台(多云)
商业：道客 DSP(如意云，交行PaaS，兴业云原生基础平台)
可观测性
跨云全局统一监控
开源：Thanos，cortex，prometheus，Graphna，opentelemetry ,https://px.dev/
商业：华为 CIE，Vmware:Tanzu Observability，红帽 EFK, prometheus
MESH
服务网格，
开源：ISTIO，kuma，OSM，SMI, kiali
商业：华为 ASM，Vmware: Service Mesh，红帽 ServiceMesh 阿里 ASM
线上支持中心
开源 SaaS：
商业：公有云都具备,PureOne
API Management
开源：APISIX，tyk
商业：红帽 3scale，Google Apigee
SaaS or Softwa
一个 site 一套
容器平台
Workload Cluster
商业：Tanzu Kubernetes Grid，CCE
Add on
APP 管理
开源: Kube Sig App, Helm Controller
商业：Vmware carvel，青云 openpitrix
监控，日志 上收，暂存
开源：prometheus，fluent-bit, metrics-server
商业：无
自动伸缩
开源: HPA，VPA ,KEDA
容器调度
开源：Kubernetes，Etcd KubeAdm，EtcdAdm coredns，npd.nfd, cloud-provider,
负载均衡
开源: Ingress Nginx，Metal LB, keeplived-vip, F5 lb，BFE, kube-vip, apiserver-network-proxy
商业：Vmware Contour/ NSX LB，红帽 ingress,router 青云 openelb
管理 UI
开源：octant,kube-dashboard LEN, backstage
商业：家家都有
管理 CLI
开源：k9s，kubectl.brew，Kui
集群 LCM
开源：cluster-api, cert-manager,kind
商业：红帽 ACM，cluster-machine, openshift-kni, gardener
AUTH/IDP
开源：dex,pinned, keyclork
商业：家家都有, Vmware Dex pinned，红帽RH-IdM
集群 运维巡检
开源：kuberhealthy, cluster-autoscaler，sonobuoy, KubeEye
商业：华为 CCE vmware sonobuoy，Crash-Diagnostics
Manage Cluster
操作系统
开源：centos 7, rocky 8，flatcar
商业：RHEL, Oracle Linux，Windows 华为 EulerOS，Vmware:Photon，红帽 CoreOS
容器引擎(CRI)
开源：docker，containerd, nvidia-gpu-support
商业：华为 Isula，Kata , Google gvisior，阿里 Pouch，红帽 podman crio
容器网络(CNI)
开源：calico,cilium,,kube-ovn, CNI-Genie，flannel，ovn-kubernetes，romana
商业：华为 yangtse，vmware：Antrea，AWS：amazon-vpc-cni-k8s，红帽 multus，openshift-sdn Egress(Antrea)
容器存储(CSI)
开源：rook，longhorn，openebs，Piraeus，local-path-provisioner，sig-storage-local-static-provisioner，nfs-subdir-external-provisioner
商业：华为 yangtse，vmware Everest，红帽 OpenShift Container Storage, LocalStorage
安装开源：kubeadm kops kubepary
一体机 裸金
开源：metal3，Sidero NVIDIA GPU Operator
商业：华为 擎天架构 阿里 神龙架构
一流公有云
3A，华为
山寨 IaaS 云
各路JB云
国产基础设施
麒麟，ARM
存储
大厂，小厂
网络
普通+SDN
非数据中心
边缘
CICD
开源：Jenkins，argo，fluxcd，flagger, concourse
商业：华为 DevCloud(PartOf)，红帽 tektoncd, Jenkins 青云 ks-jenkins
边缘
开源：, k3s，Keylime，API Server Network Proxy
商业：华为 kubeedge 阿里 ACK@Edge
DevOps
开源：未知
商业：华为 DevCloud，阿里云效
ImageBuild
开源：jib，buildah，kaniko，buildkit
商业：Tanzu build-service
Internet Redirected On-
ServerLess
开源：knative, Camel
商业：华为 functiongraph，红帽 ServerLess 阿里 FC SAE
微服务
开源：envoy，spring，Athenz，SPIRE，telepresence, jager，dapr
商业：华为 微服务引擎 CSE，Vmware：Spring Cloud, Oracle, quarkus GraalVM
阿里：MSE，CSB, EDAS，ASM
镜像仓库
开源：harbor，dragonfly
商业：华为SWR,Vmware Harbor 红帽 registry Quay 阿里 ACR
Pod aaS
开源：virtual-kubelet
商业：华为 CCI 阿里 ASK
安全
开源：opa，tuf，falco, curiefense, security-profiles-operator, Clair
商业：neuvector，aqua，华为 cgs
Chaos
开源：chaos-mesh，chaosblade, litmuschaos
批处理
开源：volcano
商业：华为 volcano
非功能:
可靠性，稳定性，兼容性
高性能，高扩展
虚拟化
开源：kubevirt
商业：红帽 kubevirt 博云 BeyondVM

-
-
-

: 2022-05-08

2.2

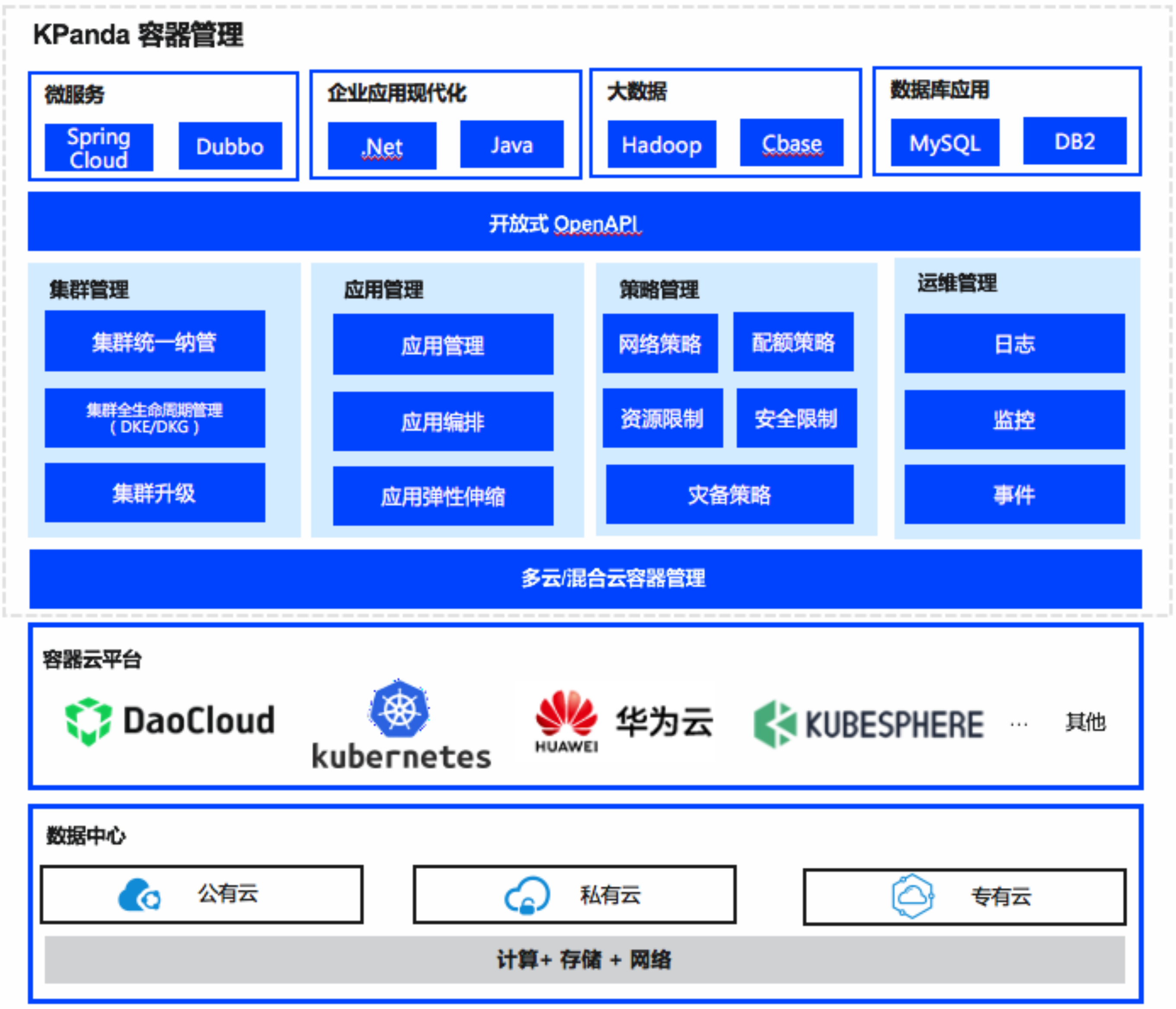
repo

(KPanda Kubernetes
Kubernetes

- Kubernetes &
- Web Kubernetes
- Kubernetes
-
-
- API Kubernetes OpenAPI

- Kubernetes
- /
-
-

- ” “
-
-
-
- Pod


KPanda 容器管理
微服务
Spring Cloud
Dubbo
企业应用现代化
.Net
Java
大数据
Hadoop
Cbase
数据库应用
MySQL
DB2
开放式 OpenAPI
集群管理
集群统一纳管
集群全生命周期管理（DKE/DKG）
集群升级
应用管理
应用管理
应用编排
应用弹性伸缩
策略管理
网络策略
配额策略
资源限制
安全限制
灾备策略
运维管理
日志
监控
事件
多云/混合云容器管理
容器云平台
DaoCloud
kubernetes
HUAWEI 华为云
KUBESPHERE
…
其他
数据中心
公有云
私有云
专有云
计算+ 存储 + 网络

: 2022-05-08

2.3

repo

-

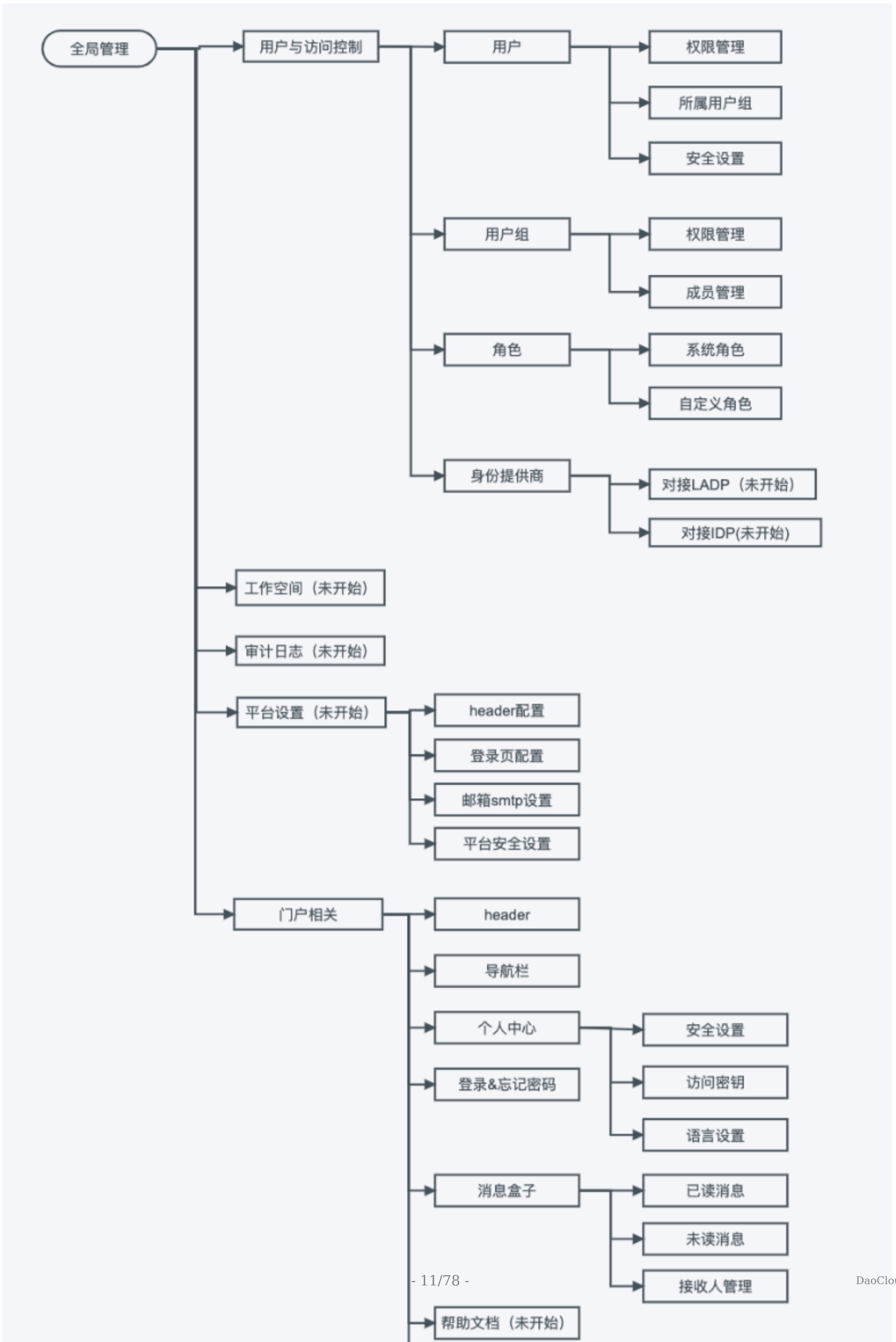
  / /

-

-

-

2.3.1


全局管理
用户与访问控制
用户
权限管理
所属用户组
安全设置
用户组
权限管理
成员管理
角色
系统角色
自定义角色
身份提供商
对接LADP（未开始）
对接IDP(未开始)
工作空间（未开始）
审计日志（未开始）
平台设置（未开始）
header配置
登录页配置
邮箱smtp设置
平台安全设置
门户相关
header
导航栏
个人中心
安全设置
访问密钥
语言设置
登录&忘记密码
消息盒子
已读消息
未读消息
接收人管理
帮助文档（未开始）

: 2022-05-08

2.4 CLUSTERPEDIA

- clusterpedia
- clusterpedi repo

Clusterpedia Wikipedia

Kubernetes OpenAPI

-
-
-
- kubernetes OpenAPI kubectl
-
-
- /
-
-

: 2022-05-08

2.5 INSIGHT

: 2022-05-08

2.6 HWAMEISTOR

: 2022-05-08

2.7 UX

: 2022-05-08

2.8

: 2022-05-08

2.9

: 2022-05-08

2.10

: 2022-05-08

2.11 MERBRIDGE

: 2022-05-08

2.12

: 2022-05-08

2.13

: 2022-05-08

2.14

: 2022-05-08

2.15 KUBE GRID

: 2022-05-08

3. PRODUCTS

3.1

/

: 2022-05-08

3.2 DCE 4.0

DCE 4.0 DaoCloud Enterprise IT DevOps IT

DCE 4.0

: 2022-05-08

3.3

DaoCloud +
VLAN SDN

: 2022-05-08

3.4

: 2022-05-08

3.5 INSIGHT

: 2022-05-08

3.6

: 2022-05-08

4. DESIGN

4.1

: 2022-05-08

4.2

: 2022-05-08

4.3

: 2022-05-08

4.4

: 2022-05-08

4.5

: 2022-05-08

5. SURVEY

5.1

https://dwiki.daocloud.io/pages/viewpage.action?pageId=87828082

: 2022-05-08

5.2 CCE

https://dwiki.daocloud.io/pages/viewpage.action?pageId=89577648

: 2022-05-08

5.3 CILIUM

: 2022-05-08

5.4 HARBOR

harbor https://dwiki.daocloud.io/pages/viewpage.action?pageId=87842865

: 2022-05-08

5.5 KARMADA

https://dwiki.daocloud.io/display/Enterprise/karmada

: 2022-05-08

5.6 KUBESPHERE

https://dwiki.daocloud.io/pages/viewpage.action?pageId=113380566

: 2022-05-08

5.7 RANCHER

https://dwiki.daocloud.io/pages/viewpage.action?pageId=89556983

: 2022-05-08

5.8

: 2022-05-08

5.9 TANZU

https://dwiki.daocloud.io/pages/viewpage.action?pageId=86277083

: 2022-05-08

6. SUPPORT

6.1

: 2022-05-08

HowTo

6.2

- Mkdocs Material
- Markdown

6.2.1

Important

`docs/` `docs/`

```
docs
├── .pages.yaml     #
├── README.md       #           README.md    default
├── SUMMARY.md
├── dce5.0          #
│   ├── 01kpanda.md #
│   ├── 02ghippo.md
│   ├── 03clusterpedia.md
│   ├── ...
├── design
│   ├── .pages.yaml #                        .pages.yaml
│   ├── README.md
│   ├── ...
├── images          #
│   ├── ghippo.png
│   ├── icon.png
│   ├── ...
├── products
│   ├── README.md
│   ├── ...
├── scaffolds       #
│   ├── .pages.yaml #         hide:true        nav
│   └── tags.md
├── stylesheets     #
│   └── extra.css
├── support
│   ├── 01-mkdocs-material.md
│   ├── README.md
│   ├── ...
└── survey
    ├── README.md
    ├── ...
```

`.pages.yaml`

`.pages.yaml`

```
title: Products     #
order: 1            #
hide: false         #
nav:                #
    - filename.md
    - filename2.md
    - ...
```

`nav` Github

```
1.   `docs/`              `dce5.0`
2.     `docs/`             `01kpanda.md`
3.
4.        gitlab ,    2
```

Front Matter

Front Matter

Tag

```yaml
tags:
    - HowTo
```

- title: Mkdocs `#`

6.2.2 Markdown

Markdown `Material` `Obsidian` `Material`

Lorem ipsum[1] dolor sit amet, consectetur adipiscing elit.[2]

Video

Inner Table

C C++

```c
#include <stdio.h>

int main(void) {
  printf("Hello world!\n");
  return 0;
}
```

```cpp
#include <iostream>

int main(void) {
  std::cout << "Hello world!" << std::endl;
  return 0;
}
```

Annotations

Phasellus posuere in sem ut cursus

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Unordered list   Ordered list

- Sed sagittis eleifend rutrum
- Donec vitae suscipit est
- Nulla tempor lobortis orci

1. Sed sagittis eleifend rutrum
2. Donec vitae suscipit est
3. Nulla tempor lobortis orci

Phasellus posuere in sem ut cursus

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Phasellus posuere in sem ut cursus

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Phasellus posuere in sem ut cursus

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Phasellus posuere in sem ut cursus

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Phasellus posuere in sem ut cursus

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Phasellus posuere in sem ut cursus

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Phasellus posuere in sem ut cursus

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Phasellus posuere in sem ut cursus

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Phasellus posuere in sem ut cursus

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Phasellus posuere in sem ut cursus

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Phasellus posuere in sem ut cursus

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Phasellus posuere in sem ut cursus

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Table

Left Center Right

Method	Description
`GET`	✓ Fetch resource
`PUT`	✓✓ Update resource
`DELETE`	✗ Delete resource

Method	Description
`GET`	✓ Fetch resource
`PUT`	✓✓ Update resource
`DELETE`	✗ Delete resource

Method	Description
`GET`	✓ Fetch resource
`PUT`	✓✓ Update resource
`DELETE`	✗ Delete resource

Charts

```
graph LR
  A[Start] --> B{Error?};
  B -->|Yes| C[Hmm...];
  C --> D[Debug];
  D --> B;
  B ---->|No| E[Yay!];
```

```
sequenceDiagram
  Alice->>John: Hello John, how are you?
  loop Healthcheck
      John->>John: Fight against hypochondria
  end
  Note right of John: Rational thoughts!
  John-->>Alice: Great!
  John->>Bob: How about you?
  Bob-->>John: Jolly good!
```

```
stateDiagram-v2
  state fork_state <<fork>>
    [*] --> fork_state
    fork_state --> State2
    fork_state --> State3

    state join_state <<join>>
    State2 --> join_state
    State3 --> join_state
    join_state --> State4
    State4 --> [*]
```

```
classDiagram
  Person <|-- Student
  Person <|-- Professor
  Person : +String name
  Person : +String phoneNumber
  Person : +String emailAddress
  Person: +purchaseParkingPass()
  Address "1" <-- "0..1" Person:lives at
  class Student{
    +int studentNumber
    +int averageMark
    +isEligibleToEnrol()
    +getSeminarsTaken()
  }
  class Professor{
    +int salary
  }
  class Address{
    +String street
    +String city
    +String state
    +int postalCode
    +String country
    -validate()
    +outputAsLabel()
  }
```

```
erDiagram
  CUSTOMER ||--o{ ORDER : places
  ORDER ||--|{ LINE-ITEM : contains
  CUSTOMER }|..|{ DELIVERY-ADDRESS : uses
```

Lorem ipsum[1] dolor sit amet, consectetur adipiscing elit.[2]

Formatting

HIGHLIGHTING CHANGES

Text can be ~~deleted~~ and replacement text added. This can also be combined into ~~one~~a single operation. Highlighting is also possible /* and comments can be added inline */.

Formatting can also be applied to blocks by putting the opening and closing tags on separate lines and adding new lines between the tags and the content.

HIGHLIGHTING TEXT

- This was marked
- This was inserted
- ~~This was deleted~~

SUB- AND SUPERSCRIPTS

- H_2O
- A^TA

ADDING KEYBOARD KEYS

^ Ctrl + ⎇ Alt + ⌦ Del

emojis

😄

Images

IMAGE

FIGCAPTION

Image caption

Lists

USING UNORDER LISTS

- Nulla et rhoncus turpis. Mauris ultricies elementum leo. Duis efficitur accumsan nibh eu mattis. Vivamus tempus velit eros, porttitor placerat nibh lacinia sed. Aenean in finibus diam.
- Duis mollis est eget nibh volutpat, fermentum aliquet dui mollis.
- Nam vulputate tincidunt fringilla.
- Nullam dignissim ultrices urna non auctor.

USING ORDERED LISTS

1. Vivamus id mi enim. Integer id turpis sapien. Ut condimentum lobortis sagittis. Aliquam purus tellus, faucibus eget urna at, iaculis venenatis nulla. Vivamus a pharetra leo.

a. Vivamus venenatis porttitor tortor sit amet rutrum. Pellentesque aliquet quam enim, eu volutpat urna rutrum a. Nam vehicula nunc mauris, a ultricies libero efficitur sed.

b. Morbi eget dapibus felis. Vivamus venenatis porttitor tortor sit amet rutrum. Pellentesque aliquet quam enim, eu volutpat urna rutrum a.

i. Mauris dictum mi lacus

ii. Ut sit amet placerat ante

iii. Suspendisse ac eros arcu

TASK LIST

- [x] orem ipsum dolor sit amet, consectetur adipiscing elit
- [ ] estibulum convallis sit amet nisi a tincidunt
- [x] n hac habitasse platea dictumst
- [x] n scelerisque nibh non dolor mollis congue sed et metus
- [ ] raesent sed risus massa
- [ ] enean pretium efficitur erat, donec pharetra, ligula non scelerisque

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit. ↩↩
2. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa. ↩↩

: 2022-05-08

6.3

- Honkit
- Hugo
- Docusaurus
- Confluence Wiki

6.3.1 DCE 5.0

-
- Kpanda
- Ghippo
- mSpider
- Insight
- Skoala

6.3.2 DCE 4.0

- DCE 4.0
- DX-Insight
-
- DNS
- Parcel
- Serverless
-

6.3.3 DSM

- DSM

6.3.4 DMP

- DMP 2.6
- DMP PoC
- DMP PoC

6.3.5

- KLTS
- Clusterpedia
- Merbridge
- HwameiStor

: 2022-05-08

6.4

: 2022-05-08

6.5

idea → → → → → → →

: 2022-05-08

6.6

: 2022-05-08

6.7

06frontend-ux.md

: 2022-05-08

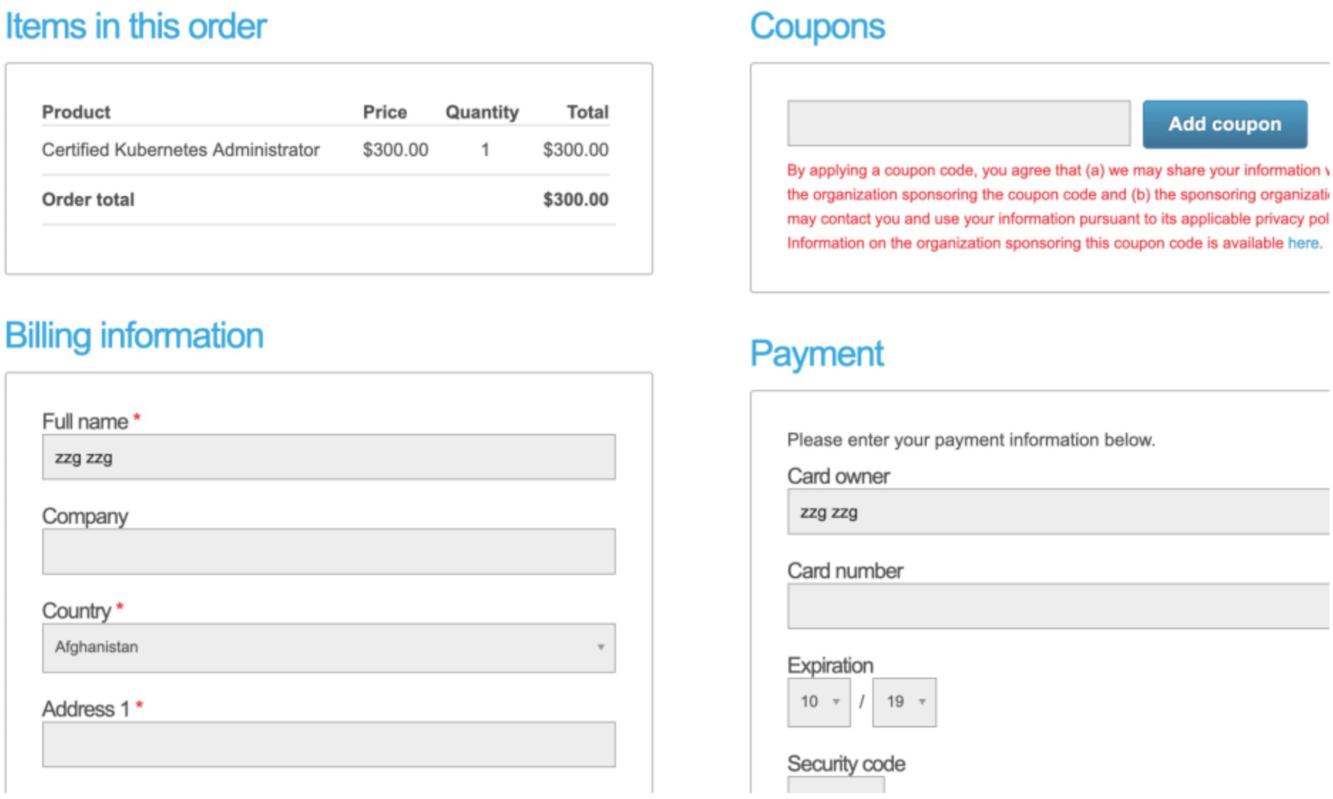
6.8 CKA

6.8.1 DaoCloud CKA

Info

1.
2. wiki
3.

1. https://identity.linuxfoundation.org/user/login
2. https://www.cncf.io/certification/CKA/ * “ **REGISTER FOR THE EXAM**” *



Items in this order

Product	Price	Quantity	Total
Certified Kubernetes Administrator	$300.00	1	$300.00
Order total			**$300.00**

Coupons

Add coupon

By applying a coupon code, you agree that (a) we may share your information w
the organization sponsoring the coupon code and (b) the sponsoring organizati
may contact you and use your information pursuant to its applicable privacy pol
Information on the organization sponsoring this coupon code is available here.

Billing information

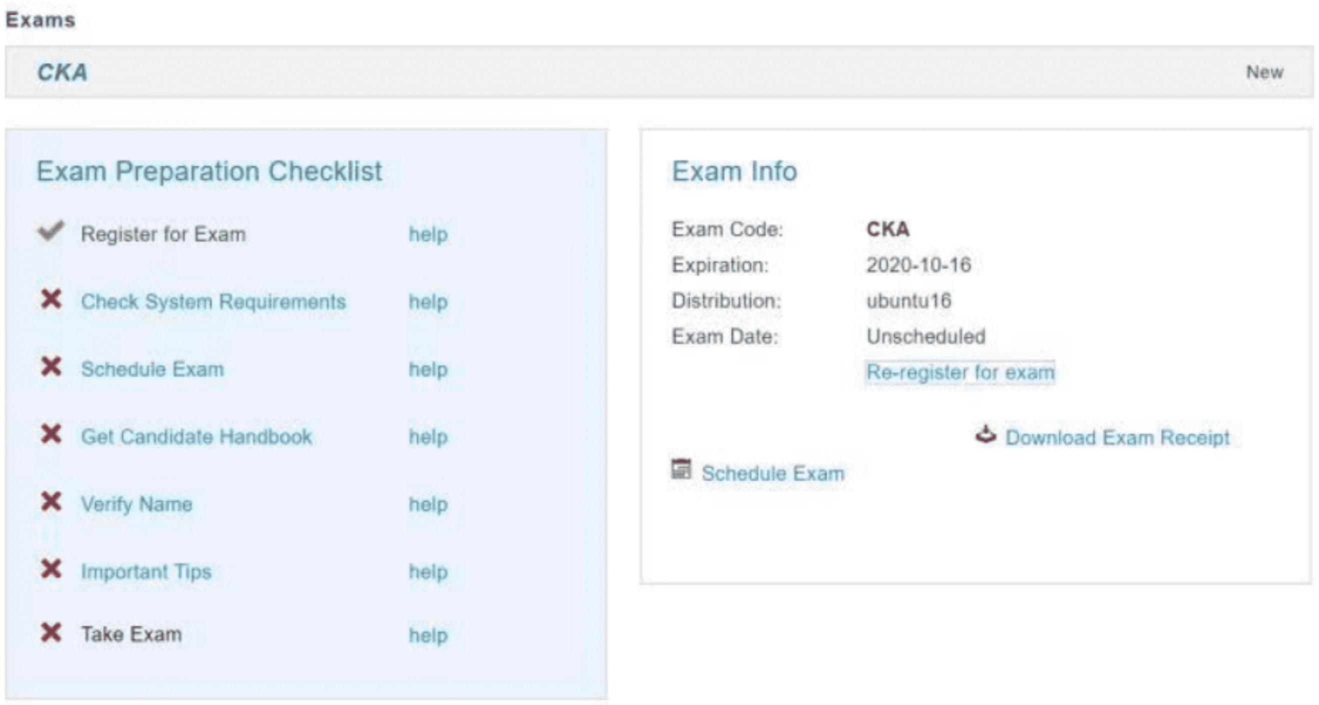
Full name *	zzg zzg
Company	
Country *	Afghanistan
Address 1 *	

Payment

Please enter your payment information below.

Card owner	zzg zzg
Card number	
Expiration	10 / 19
Security code	

1. Coupons( ) DCUBEOFFER 48 , (visa)
2. https://portal.linuxfoundation.org/portal ( ) "checklist"



Exams

CKA New

Exam Preparation Checklist

✓	Register for Exam	help
✗	Check System Requirements	help
✗	Schedule Exam	help
✗	Get Candidate Handbook	help
✗	Verify Name	help
✗	Important Tips	help
✗	Take Exam	help

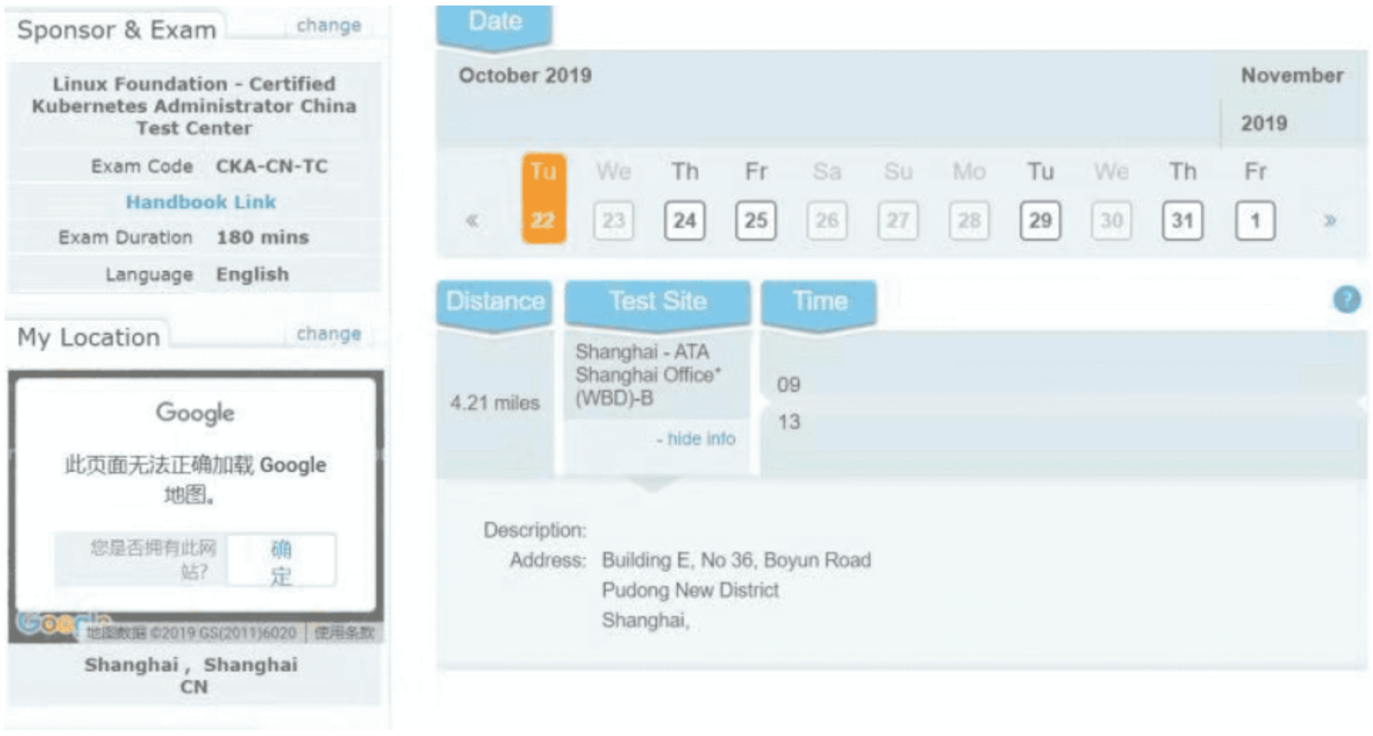
Exam Info

Exam Code:	CKA
Expiration:	2020-10-16
Distribution:	ubuntu16
Exam Date:	Unscheduled
	Re-register for exam

Download Exam Receipt

Schedule Exam

1. "schedule Exam" -- 24



Sponsor & Exam change

Linux Foundation - Certified Kubernetes Administrator China Test Center

Exam Code	CKA-CN-TC
	Handbook Link
Exam Duration	180 mins
Language	English

My Location change

Google
此页面无法正确加载 Google 地图。
您是否拥有此网站?
确定
地图数据 ©2019 GS(2011)6020 | 使用条款

Shanghai , Shanghai CN

Date

October 2019 November 2019

	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	
«	22	23	24	25	26	27	28	29	30	31	1	»

Distance	Test Site	Time
4.21 miles	Shanghai - ATA Shanghai Office* (WBD)-B - hide info	09 13

Description:
Address: Building E, No 36, Boyun Road
Pudong New District
Shanghai,

- https://dwiki.daocloud.io/pages/viewpage.action?pageId=13060398
- https://dwiki.daocloud.io/pages/viewpage.action?pageId=16104222
- CKA https://dwiki.daocloud.io/pages/viewpage.action?pageId=55337980

: 2022-05-08

6.8.2 CKA

- CKA 500Kbps 256Kbps
-
- `Innovative Exams Screensharing`
- CKA
- https://training.linuxfoundation.cn/certificate/details/1
- 2088
- CKA
- – (CKA)
- – (CKA-CN)
- =>

CKA

CKA

https://training.linuxfoundation.cn/certificate/details/1

2088

CKA – (CKA) – (CKA-CN)

CKA

-
- 24
- WebDelivery Compatibility Check

CKA

1. chrome chromium innovactive exams screensharing cookie chrome
2. –
3.
4.

CKA

-
-
-
-
-
-
-
-

-
-
-
-

- 4,5
- 50
- kubeenetes.io
- 36
- CKA 3 3

:

- https://www.cnblogs.com/miketwais/p/CKA.html
- https://mp.weixin.qq.com/s/pK-_nTAQoqNwNVHhiqCEjQ ( )

- https://www.bilibili.com/video/BV1S7411m7vM

: 2022-05-08

6.8.3 CKA

KUBERNETES

- Kubernetes
- Kubernetes
- kubeadm
- etcd
- kubectl shell

KUBERNETES HELM

- Pod
-
- Deployment
- Deployment / /
- StatefulSet
- DaemonSet
-
- ConfigMaps Secrets
- Kubernetes
- Pod
-
-
- Helm / / /
- Helm Chart

KUBERNETES

- Service Endpoint
- Service Iptables IPVS
- Service ClusterIP
- NodePort Ingress LoadBalancer
- CoreDNS
- CoreDNS
- Pod/ Node/ Node/
-
-

KUBERNETES

- Volume PV PVC StorageClass
-
-
-

-
- RBAC
- Pod
- Network Policy

KUBERNETES

- Kubenetes
-
-
-
-
- Kubernetes

: 2022-05-08

6.8.4 CKA -01

- CKA 500Kbps 256Kbps
-
- `Innovative Exams Screensharing`
- CKA
- ,

1

```
**Set configuration context $kubectl config use-context k8s. Monitor the logs of Pod foobar and Extract log lines corresponding to error unable-to-access-
website . Write them to /opt/KULM00612/foobar.**

          $kubectl config use context k8s   Pod foobar         “unable-to-access-website”           /opt/KULM00612/foobar

         pod


        k8s
kubectl config use context k8s
[root@k8s-node1 ~]# kubectl   logs nginx|grep "unable-to-access-website" > /tmp/task.txt
```

2

```
**Set configuration context $kubectl config use-context k8s. List all PVs sorted by capacity, saving the full kubectl output to /opt/KUCC0006/my_volumes. Use
kubectl own functionally for sorting the output, and do not manipulate it any further**

          $kubectl config use context k8s           PV     kubectl     /opt/KUCC0006/my_volumes       kubectl
   pv


[root@k8s-node1 pv]# kubectl  get pv -A  --sort-by={.spec.capacity.sotrge} >  /opt/cka-tak.txt


[root@k8s-node1 pv]# cat  /opt/cka-tak.txt
NAME     CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
nfspv1   1Gi        RWO            Recycle          Available           mynfs                   4m12s
[root@k8s-node1 pv]#
```

3

```
**Set configuration context $kubectl config use-context k8s. Ensure a single instance of Pod nginx is running on each node of the Kubernetes cluster where
nginx also represents the image name which has to be used. Do no override any taints currently in place. Use Daemonset to complete this task and use
ds.kusc00612 as Daemonset name**

          $kubectl config use context k8s    Kubernetes          Pod nginx        nginx                            Daemonset        ds.kusc00612

          Daemonset
https://kubernetes.io/zh/docs/concepts/workloads/controllers/daemonset/

[root@k8s-node1 ~]# vim Daemonset.ayml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: ds.kusc00612
  labels:
    k8s-app: ds.kusc00612
spec:
  selector:
    matchLabels:
      name: ds.kusc00612
  template:
    metadata:
      labels:
        name: ds.kusc00612
    spec:
      containers:
      - name: nginx
        image: nginx

apply  yaml
[root@k8s-node1 ~]# kubectl  apply -f Daemonset.ayml
daemonset.apps/ds.kusc00612 created
[root@k8s-node1 ~]# kubectl  get  daemonset
```

```
NAME           DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
ds.kusc00612   1         1         1       1            1           <none>          2m14s
```

4

```
Set configuration context $kubectl config use-context k8s Perform the following tasks: Add an init container to lumpy-koala(which has been defined in spec
file /opt/kucc00100/pod-specKUCC00612.yaml). The init container should create an empty file named /workdir/calm.txt. If /workdir/calm.txt is not detected,
the Pod should exit. Once the spec file has updated with the init container definition, the Pod should be created
          init     lumpy-koala     /opt/kucc00100/pod-specKUCC00612.yaml     init          /workdir/calm.txt    .   /workdir/calm.txt     Pod       init
spec        Pod
      /opt/kucc00100/pod-specKUCC00612.yaml         Yaml                       kubectl get po | grep pod           pod          Yaml,  apply.   Initcontainer,
        /workdir                       empDir         liveness

        :
https://kubernetes.io/docs/concepts/workloads/pods/init-containers/#using-init-containers
https://kubernetes.io/docs/concepts/storage/volumes/#emptydir
https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/
[root@k8s-node1 ~]# vim  init-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox:1.28
    command: ['sh', '-c', 'echo The app is running! && sleep 3600']
  initContainers:
  - name: init-myservice
    image: busybox:1.28
    command: ['sh', '-c', "touch /workdir/calm.txt"]
    volumeMounts:
    - mountPath: /workdir
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}

[root@k8s-node1 ~]# kubectl  apply -f  init-pod.yaml
pod/myapp-pod created
```

5

```
Set configuration context $kubectl config use-context k8s. Create a pod named kucc6 with a single container for each of the following images running
inside(there may be between 1 and 4 images specified):nginx +redis+memcached+consul
        kucc6 pod                        1 4     nginx+redis+memcached+consur


     :https://kubernetes.io/zh/docs/concepts/scheduling-eviction/assign-pod-node/

[root@k8s-node1 ~]# vim images.yaml
apiVersion: v1
kind: Pod
metadata:
  name: kucc6
  labels:
    env: kucc6
spec:
  containers:
  - name: nginx
    image: nginx
  - name: redis
    image: redis
  - name: memcached
    image: memcached
  - name: consul
    image: consul


[root@k8s-node1 ~]# kubectl  apply -f images.yaml
pod/kucc6 created
```

6

```
Set configuration context $kubectl config use-context k8s Schedule a Pod as follows: Name: nginxkusc00612 Image: nginx Node selector: disk=ssd
      Pod     nginx     nginx     label disk=ssd node
   pod        Nodeselector

        nodeselector yaml  ,
    https://kubernetes.io/zh/docs/concepts/scheduling-eviction/assign-pod-node/
#     node              lable
[root@k8s-node1 ~]# kubectl  get nodes   --show-labels
[root@k8s-node1 ~]# vim nodeSelector.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginxkusc00612
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  nodeSelector:
    disktype: ssd

[root@k8s-node1 ~]# kubectl  apply -f nodeSelector.yaml
pod/nginxkusc00612 created

[root@k8s-node1 ~]# kubectl  get pods nginxkusc00612
NAME             READY   STATUS    RESTARTS   AGE
nginxkusc00612   1/1     Running   0          3m
```

7

```
Set configuration context $kubectl config use-context k8s. Create a deployment as follows: Name: nginxapp Using container nginx with version 1.11.9-alpine.
The deployment should contain 3 replicas. Next, deploy the app with new version 1.12.0-alpine by performing a rolling update and record that
update.Finally,rollback that update to the previous version 1.11.9-alpine.
     deploy,
    Deployment

     https://kubernetes.io/zh/docs/concepts/workloads/controllers/deployment/
[root@k8s-node1 ~]# kubectl  create deployment  nginxapp --image=nginx:1.11.9-alpine
deployment.apps/nginxapp created
[root@k8s-node1 ~]# kubectl  scale     deployment   nginxapp  --replicas=3
deployment.apps/nginxapp scaled

[root@k8s-node1 ~]# kubectl set image  deployment/nginxapp nginx=nginx:1.12.0-alpine --record=true
deployment.apps/nginxapp image updated
[root@k8s-node1 ~]# kubectl  rollout  undo   deployment.apps/nginxapp
```

8

```
Set configuration context $kubectl config use-context k8s Create and configure the service front-endservice so it's accessible through NodePort/ClusterIp and
routes to the existing pod named nginxkusc00612
     service        pod: nginxkusc00612

    https://kubernetes.io/zh/docs/tasks/access-application-cluster/connecting-frontend-backend/

#                   Endpoints
[root@k8s-node1 ~]# cat service.yaml
apiVersion: v1
kind: Service
metadata:
  name: pod-service
spec:
  selector:
    app: front-end
  type: NodePort
  ports:
  - protocol: TCP
    port: 80
    targetPort: http
```

9

```
Set configuration context $kubectl config use-context k8s Create a Pod as follows: Name: jenkins Using image: jenkins In a new Kubernetes namespace named pro-
test
             jenkins pod


       pro-test
[root@k8s-node1 ~]# kubectl  get namespaces  pro-test


[root@k8s-node1 ~]# kubectl    run   jenkins   --image=jenkins  --namespace=pro-test


[root@k8s-node1 ~]# kubectl  get pods -n pro-test
```

10

```
Set configuration context $kubectl config use-context k8s Create a deployment spec file that will: Launch 7 replicas of the redis image with the label :
app_enb_stage=dev Deployment name: kual00612 Save a copy of this spec file to /opt/KUAL00612/deploy_spec.yaml (or .json) When you are done,clean up(delete)
any new k8s API objects that you produced during this task

          $kubectl config use context k8s                 redis   7        app_enb_stage=dev deployment name:kual00612           /opt/kual00612/
deploy_spec.yaml  .json                        k8sapi
     7   redis deploy         yaml

    https://kubernetes.io/docs/concepts/workloads/controllers/deployment/
[root@k8s-node1 ~]# vim ReplicaSet.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kual00612
  labels:
    app_enb_stage: dev
spec:
  replicas: 7
  selector:
    matchLabels:
      app: kual00612
  template:
    metadata:
      labels:
        app: kual00612
    spec:
      containers:
      - name: redis
        image: redis


[root@k8s-node1 ~]# kubectl  get pods |grep kual00612|grep  Running|wc -l

    yaml   /opt/KUAL00612/deploy_spec.yaml
cat ReplicaSet.yaml >  /opt/KUAL00612/deploy_spec.yaml
```

11

```
Set configuration context $kubectl config use-context k8s Create a file /opt/KUCC00612/kucc00612.txt that lists all pods that implement Service foo in
Namespace production. The format of the file should be one pod name per line.
     foo service     pod

kubecet get svc   -n production  --show-lables|grep foo
kubectl  get pods -nccod45 -l name=foo |grep -v NAME|awk '{print $1}' >>   /opt/KUCC00302/kucc00302.txt
```

12

```
Set configuration context $kubectl config use-context k8s Create a Kubernetes Secret as follows: Name: super-secret credential: blob, Create a Pod named pod-
secrets-via-file using the redis image which mounts a secret named super-secret at /secrets. Create a second Pod named pod-secretsvia-env using the redis
image, which exports credential as
       Kubernetes Secret      Name:super Secret credential:blob   redis        Pod secrets Pod     /secrets       super Secret      redis         Pod
secretsvia env Pod
     secret   pod   Volume        secret

   https://kubernetes.io/zh/docs/concepts/configuration/secret/
[root@k8s-node1 ~]#  echo blob |base64
YmxvYgo=
```

13

```
Set configuration context $kubectl config use-context k8s Create a pod as follows: Name: nonpersistent-redis Container image: redis Named-volume with name:
cache-control Mount path : /data/redis It should launch in the pre-prod namespace and the volume MUST NOT be persistent.
       pod    volume

    https://kubernetes.io/zh/docs/concepts/storage/volumes/
[root@k8s-node1 ~]# cat volumes.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nonpersistent-redis
spec:
  containers:
  - image: redis
    name: nonpersistent-redis
    volumeMounts:
    - mountPath: /data/redis
      name: cache-control
  volumes:
  - name: cache-control
    emptyDir: {}
```

14

```
Set configuration context $kubectl config use-context k8s Scale the deployment webserver to 6 pods


   https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#scale
[root@k8s-node1 ~]# kubectl  scale --replicas=6 webserver
```

15

```
Set configuration context $kubectl config use-context k8s Check to see how many nodes are ready (not including nodes tainted NoSchedule) and write the number
to /opt/nodenum.
         ready          NoSchedule
kubectl describe nodes `kubectl get nodes|grep Ready|awk '{print $1}'` | grep Taints|grep -vc NoSchedule >  /opt/nodenum
```

16

```
Set configuration context $kubectl config use-context k8s Create a deployment as follows: Name: nginxdns Exposed via a service : nginx-dns Ensure that the
service & pod are accessible via their respective DNS records The container(s) within any Pod(s) running as a part of this deployment should use the nginx
image. Next, use the utility nslookup to look up the DNS records of the service & pod and write the output to /opt/service.dns and /opt/pod.dns respectively.
Ensure you use the busybox:1.28 image (or earlier) for any testing, an the latest release has an upstream bug which impacts the use of nslookup
     service deployment     service dns pod dns

        :https://kubernetes.io/docs/tasks/access-application-cluster/connecting-frontend-backend/
      https://kubernetes.io/zh/docs/concepts/workloads/pods/init-containers/

[root@k8s-node1 dns]# cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend
spec:
  selector:
    matchLabels:
      app: nginxdns
  replicas: 1
  template:
    metadata:
      labels:
        app: nginxdns
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - name: http
              containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginxdns
spec:
  selector:
    app: nginxdns
  ports:
  - protocol: TCP
    port: 80
    targetPort: http
---
apiVersion: v1
kind: Pod
metadata:
  name: busybox-test
  labels:
    app: busybox-test
spec:
  containers:
  - name: myapp-container
    image: busybox:1.28
    command: ['sh', '-c', 'echo The app is running! && sleep 3600']

[root@k8s-node1 dns]#  kubectl exec -ti busybox-test -- nslookup nginxdns   #  svc   >/opt/service.dns

kubectl exec -ti busybox-test -- nslookup  10.244.1.52   > /opt/pod.dns#Pod ip
```

17

```
No configuration context change required for this item Create a snapshot of the etcd instance running at https://127.0.0.1:2379 saving the snapshot to the
file path /data/backup/etcd-snapshot.db The etcd instance is running etcd version 3.2.18 The following TLS certificates/key are supplied for connecting to the
server with etcdctl CA certificate: /opt/KUCM0612/ca.crt Client certificate: /opt/KUCM00612/etcdclient.crt Client key: /opt/KUCM00612/etcd-client.key

   https://kubernetes.io/zh/docs/tasks/administer-cluster/configure-upgrade-etcd/
  etcd
ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 \
  --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/peer.crt --key=/etc/kubernetes/pki/etcd/peer.key \
  snapshot save /etc/kubernetes/pki/etcd/etcd-snapshot-test.db

  etcd
TCDCTL_API=3 etcdctl --write-out=table snapshot status /etc/kubernetes/pki/etcd/etcd-snapshot-test.db

  etcd
  ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379  snapshot restore \
  --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/peer.crt --key=/etc/kubernetes/pki/etcd/peer.key\
  /etc/kubernetes/pki/etcd/etcd-snapshot-test.db

      etcd      nodes pods
```

18

```
Set configuration context $kubectl config use-context ek8s Set the node labelled with name=ek8s-node-1 as unavailable and reschedule all the pods running on
it.
      name=ek8s-node-1              pod               kubectl drain

kubectl cordon  ek8s-node-1  #
kubectl drain ek8s-node-1 --ignore-daemonsets --delete-local-data --force
        get nodes
```

19

```
Set configuration context $kubectl config use-context wk8s A Kubernetes worker node,labelled with name=wk8s-node-0 is in state NotReady. Investigate why this
is the case, and perform any appropriate steps to bring the node to a Ready state, Ensuring that any changes are made permanent. Hints: You can ssh to the
failed node using $ssh wk8s-node-0. You can assume elevated privileges on the node with the following command $sudo -i
     wk8s-node-0 NotReady             ready
```

20

```
21 Set configuration context $kubectl config use-context wk8s Configure the kubelet system managed service,on the node labelled with name=wk8s-node-1, to
Launch a Pod containing a single container of image nginx named myservice automatically. Any spec files required should be placed in the /etc/kubernetes/
manifests directory on the node. Hints: You can ssh to the failed node using $ssh wk8snode-1. You can assume elevated privileges on the node with the
following command $sudo -i
```

21

```
         master1 node1    admin.conf
```

22

```
Set configuration context $kubectl configuse-context bk8s Given a partially-functioning Kubernetes cluser, identify symptoms of failure on the cluter.
Determine the node, the failing service and take actions to bring up the failed service and restore the health of the cluser. Ensure that any changes are
made permanently. The worker node in this cluster is labelled with name=bk8s-node-0 Hints: You can ssh to the relevant nodes using $ssh $(NODE) where $(NODE)
is one of bk8s-master-0 or bk8s-node-0. You can assume elevated privileges on any node in the cluster with the following command: $ sudo -i.
```

23

```
Set configuration context $kubectl config use-context hk8s Create a persistent volume with name appconfig of capacity 1Gi and access mode ReadWriteMany. The
type of volume is hostPath and its locationis /srv/app-config
```

CKA

1

```
Task weight: 1%

You have access to multiple clusters from your main terminal through kubectl contexts. Write all those context names into /opt/course/1/contexts.

Next write a command to display the current context into /opt/course/1/context_default_kubectl.sh, the command should use kubectl.

Finally write a second command doing the same thing into /opt/course/1/context_default_no_kubectl.sh, but without the use of kubectl.
```

2

```
Task weight: 3%

Use context: kubectl config use-context k8s-c1-H

Create a single Pod of image httpd:2.4.41-alpine in Namespace default. The Pod should be named pod1 and the container should be named pod1-container. This
Pod should only be scheduled on a master node, do not add new labels any nodes.

Shortly write the reason on why Pods are by default not scheduled on master nodes into /opt/course/2/master_schedule_reason .
```

3

```
Use context: kubectl config use-context k8s-c1-H

There are two Pods named o3db-* in Namespace project-c13. C13 management asked you to scale the Pods down to one replica to save resources. Record the action.
```

4

```
Task weight: 1%

Use context: kubectl config use-context k8s-c1-H

There are two Pods named o3db-* in Namespace project-c13. C13 management asked you to scale the Pods down to one replica to save resources. Record the action.
```

5

```
Task weight: 4%

Use context: kubectl config use-context k8s-c1-H

Do the following in Namespace default. Create a single Pod named ready-if-service-ready of image nginx:1.16.1-alpine. Configure a LivenessProbe which simply
runs true. Also configure a ReadinessProbe which does check if the url http://service-am-i-ready:80 is reachable, you can use wget -T2 -O- http://service-am-
```

```
i-ready:80 for this. Start the Pod and confirm it isn't ready because of the ReadinessProbe.

Create a second Pod named am-i-ready of image nginx:1.16.1-alpine with label id: cross-server-ready. The already existing Service service-am-i-ready should
now have that second Pod as endpoint.

Now the first Pod should be in ready state, confirm that.
```

6

```
Task weight: 1%

Use context: kubectl config use-context k8s-c1-H

There are various Pods in all namespaces. Write a command into /opt/course/5/find_pods.sh which lists all Pods sorted by their AGE
(metadata.creationTimestamp).

Write a second command into /opt/course/5/find_pods_uid.sh which lists all Pods sorted by field metadata.uid. Use kubectl sorting for both commands.
```

7

```
Task weight: 8%

Use context: kubectl config use-context k8s-c1-H

Create a new PersistentVolume named safari-pv. It should have a capacity of 2Gi, accessMode ReadWriteOnce, hostPath /Volumes/Data and no storageClassName
defined.

Next create a new PersistentVolumeClaim in Namespace project-tiger named safari-pvc . It should request 2Gi storage, accessMode ReadWriteOnce and should not
define a storageClassName. The PVC should bound to the PV correctly.

Finally create a new Deployment safari in Namespace project-tiger which mounts that volume at /tmp/safari-data. The Pods of that Deployment should be of
image httpd:2.4.41-alpine.
```

8

```
Task weight: 2%

Use context: kubectl config use-context k8s-c1-H

Ssh into the master node with ssh cluster1-master1. Check how the master components kubelet, kube-apiserver, kube-scheduler, kube-controller-manager and etcd
are started/installed on the master node. Also find out the name of the DNS application and how it's started/installed on the master node.

Write your findings into file /opt/course/8/master-components.txt. The file should be structured like:

# /opt/course/8/master-components.txt
kubelet: [TYPE]
kube-apiserver: [TYPE]
kube-scheduler: [TYPE]
kube-controller-manager: [TYPE]
etcd: [TYPE]
dns: [TYPE] [NAME]
Choices of [TYPE] are: not-installed, process, static-pod, pod
```

9

```
Task weight: 5%

Use context: kubectl config use-context k8s-c2-AC

Ssh into the master node with ssh cluster2-master1. Temporarily stop the kube-scheduler, this means in a way that you can start it again afterwards.

Create a single Pod named manual-schedule of image httpd:2.4-alpine, confirm its started but not scheduled on any node.

Now you're the scheduler and have all its power, manually schedule that Pod on node cluster2-master1. Make sure it's running.

Start the kube-scheduler again and confirm its running correctly by creating a second Pod named manual-schedule2 of image httpd:2.4-alpine and check if it's
running on cluster2-worker1.
```

10

```
Use context: kubectl config use-context k8s-c1-H

Create a new ServiceAccount processor in Namespace project-hamster. Create a Role and RoleBinding, both named processor as well. These should allow the new
SA to only create Secrets and ConfigMaps in that Namespace.
```

11

```
Use context: kubectl config use-context k8s-c1-H

Use Namespace project-tiger for the following. Create a DaemonSet named ds-important with image httpd:2.4-alpine and labels id=ds-important and
uuid=18426a0b-5f59-4e10-923f-c0e078e82462. The Pods it creates should request 10 millicore cpu and 10 megabytes memory. The Pods of that DaemonSet should run
on all nodes.
```

12

```
Use context: kubectl config use-context k8s-c1-H

Use Namespace project-tiger for the following. Create a Deployment named deploy-important with label id=very-important (the pods should also have this label)
and 3 replicas. It should contain two containers, the first named container1 with image nginx:1.17.6-alpine and the second one named container2 with image
kubernetes/pause.

There should be only ever one Pod of that Deployment running on one worker node. We have two worker nodes: cluster1-worker1 and cluster1-worker2. Because the
Deployment has three replicas the result should be that on both nodes one Pod is running. The third Pod won't be scheduled, unless a new worker node will be
added.

In a way we kind of simulate the behaviour of a DaemonSet here, but using a Deployment and a fixed number of replicas.
```

13

```
Create a Pod named multi-container-playground in Namespace default with three containers, named c1, c2 and c3. There should be a volume attached to that Pod
and mounted into every container, but the volume shouldn't be persisted or shared with other Pods.

Container c1 should be of image nginx:1.17.6-alpine and have the name of the node where its Pod is running on value available as environment variable
MY_NODE_NAME.

Container c2 should be of image busybox:1.31.1 and write the output of the date command every second in the shared volume into file date.log. You can use
while true; do date >> /your/vol/path/date.log; sleep 1; done for this.

Container c3 should be of image busybox:1.31.1 and constantly write the content of file date.log from the shared volume to stdout. You can use tail -f /your/
vol/path/date.log for this.

Check the logs of container c3 to confirm correct setup.
```

14

```
Use context: kubectl config use-context k8s-c1-H

You're ask to find out following information about the cluster k8s-c1-H:

How many master nodes are available?
How many worker nodes are available?
What is the Service CIDR?
Which Networking (or CNI Plugin) is configured and where is its config file?
Which suffix will static pods have that run on cluster1-worker1?
Write your answers into file /opt/course/14/cluster-info, structured like this:

# /opt/course/14/cluster-info
1: [ANSWER]
2: [ANSWER]
3: [ANSWER]
4: [ANSWER]
5: [ANSWER]
```

15

```
Use context: kubectl config use-context k8s-c2-AC

Write a command into /opt/course/15/cluster_events.sh which shows the latest events in the whole cluster, ordered by time. Use kubectl for it.

Now kill the kube-proxy Pod running on node cluster2-worker1 and write the events this caused into /opt/course/15/pod_kill.log.

Finally kill the containerd container of the kube-proxy Pod on node cluster2-worker1 and write the events into /opt/course/15/container_kill.log.

Do you notice differences in the events both actions caused?
```

16

```
Use context: kubectl config use-context k8s-c1-H

Create a new Namespace called cka-master.

Write the names of all namespaced Kubernetes resources (like Pod, Secret, ConfigMap...) into /opt/course/16/resources.txt.

Find the project-* Namespace with the highest number of Roles defined in it and write its name and amount of Roles into /opt/course/16/crowded-namespace.txt.
```

17

```
Use context: kubectl config use-context k8s-c1-H

In Namespace project-tiger create a Pod named tigers-reunite of image httpd:2.4.41-alpine with labels pod=container and container=pod. Find out on which node
the Pod is scheduled. Ssh into that node and find the containerd container belonging to that Pod.

Using command crictl:

Write the ID of the container and the info.runtimeType into /opt/course/17/pod-container.txt

Write the logs of the container into /opt/course/17/pod-container.log
```

18

```
Use context: kubectl config use-context k8s-c3-CCC

There seems to be an issue with the kubelet not running on cluster3-worker1. Fix it and confirm that cluster3 has node cluster3-worker1 available in Ready
state afterwards. Schedule a Pod on cluster3-worker1.

Write the reason of the is issue into /opt/course/18/reason.txt.
```

19

```
this task can only be solved if questions 18 or 20 have been successfully implemented and the k8s-c3-CCC cluster has a functioning worker node

Use context: kubectl config use-context k8s-c3-CCC

Do the following in a new Namespace secret. Create a Pod named secret-pod of image busybox:1.31.1 which should keep running for some time, it should be able
to run on master nodes as well.

There is an existing Secret located at /opt/course/19/secret1.yaml, create it in the secret Namespace and mount it readonly into the Pod at /tmp/secret1.

Create a new Secret in Namespace secret called secret2 which should contain user=user1 and pass=1234. These entries should be available inside the Pod's
container as environment variables APP_USER and APP_PASS.

Confirm everything is working.
```

20

```
Your coworker said node cluster3-worker2 is running an older Kubernetes version and is not even part of the cluster. Update kubectl and kubeadm to the
version that's running on cluster3-master1. Then add this node to the cluster, you can use kubeadm for this.
```

21

```
Use context: kubectl config use-context k8s-c3-CCC

Create a Static Pod named my-static-pod in Namespace default on cluster3-master1. It should be of image nginx:1.16-alpine and have resource requests for 10m
CPU and 20Mi memory.

Then create a NodePort Service named static-pod-service which exposes that static Pod on port 80 and check if it has Endpoints and if its reachable through
the cluster3-master1 internal IP address. You can connect to the internal node IPs from your main terminal.
```

22

```
Use context: kubectl config use-context k8s-c2-AC

Check how long the kube-apiserver server certificate is valid on cluster2-master1. Do this with openssl or cfssl. Write the exipiration date into /opt/course/
22/expiration.

Also run the correct kubeadm command to list the expiration dates and confirm both methods show the same date.

Write the correct kubeadm command that would renew the apiserver server certificate into /opt/course/22/kubeadm-renew-certs.sh.
```

23

```
Use context: kubectl config use-context k8s-c2-AC

Node cluster2-worker1 has been added to the cluster using kubeadm and TLS bootstrapping.

Find the "Issuer" and "Extended Key Usage" values of the cluster2-worker1:

kubelet client certificate, the one used for outgoing connections to the kube-apiserver.
kubelet server certificate, the one used for incoming connections from the kube-apiserver.
Write the information into file /opt/course/23/certificate-info.txt.

Compare the "Issuer" and "Extended Key Usage" fields of both certificates and make sense of these.
```

24

```
Use context: kubectl config use-context k8s-c1-H

There was a security incident where an intruder was able to access the whole cluster from a single hacked backend Pod.

To prevent this create a NetworkPolicy called np-backend in Namespace project-snake. It should allow the backend-* Pods only to:

connect to db1-* Pods on port 1111
connect to db2-* Pods on port 2222
Use the app label of Pods in your policy.

After implementation, connections from backend-* Pods to vault-* Pods on port 3333 should for example no longer work.
```

25

```
Use context: kubectl config use-context k8s-c3-CCC
```

```
Make a backup of etcd running on cluster3-master1 and save it on the master node at /tmp/etcd-backup.db.

Then create a Pod of your kind in the cluster.

Finally restore the backup, confirm the cluster is still working and that the created Pod is no longer with us.
```

26

```
Use context: kubectl config use-context k8s-c1-H

Check all available Pods in the Namespace project-c13 and find the names of those that would probably be terminated first if the Nodes run out of resources (cpu
or memory) to schedule all Pods. Write the Pod names into /opt/course/e1/pods-not-stable.txt.
```

27

```
Use context: kubectl config use-context k8s-c1-H

There is an existing ServiceAccount secret-reader in Namespace project-hamster. Create a Pod of image curlimages/curl:7.65.3 named tmp-api-contact which uses
this ServiceAccount. Make sure the container keeps running.

Exec into the Pod and use curl to access the Kubernetes Api of that cluster manually, listing all available secrets. You can ignore insecure https
connection. Write the command(s) for this into file /opt/course/e4/list-secrets.
```

26

```
Preview Question 1
Use context: kubectl config use-context k8s-c2-AC

The cluster admin asked you to find out the following information about etcd running on cluster2-master1:

Server private key location
Server certificate expiration date
Is client certificate authentication endabled
Write these information into /opt/course/p1/etcd-info.txt

Finally you're asked to save an etcd snapshot at /etc/etcd-snapshot.db on cluster2-master1 and display its status.
```

27

```
Preview Question 2
Use context: kubectl config use-context k8s-c1-H

You're asked to confirm that kube-proxy is running correctly on all nodes. For this perform the following in Namespace project-hamster:

Create a new Pod named p2-pod with two containers, one of image nginx:1.21.3-alpine and one of image busybox:1.31. Make sure the busybox container keeps
running for some time.

Create a new Service named p2-service which exposes that Pod internally in the cluster on port 3000->80.

Find the kube-proxy container on all nodes cluster1-master1, cluster1-worker1 and cluster1-worker2 and make sure that it's using iptables. Use command crictl
for this.

Write the iptables rules of all nodes belonging the created Service p2-service into file /opt/course/p2/iptables.txt.

Finally delete the Service and confirm that the iptables rules are gone from all nodes.
```

26

```
Preview Question 3
Use context: kubectl config use-context k8s-c2-AC

Create a Pod named check-ip in Namespace default using image httpd:2.4.41-alpine. Expose it on port 80 as a ClusterIP Service named check-ip-service.
Remember/output the IP of that Service.

Change the Service CIDR to 11.96.0.0/12 for the cluster.

Then create a second Service named check-ip-service2 pointing to the same Pod to check if your settings did take effect. Finally check if the IP of the first
Service has changed.
```

CKA

-
- etcd k8s
- k8s k8s
- : https://blog.csdn.net/u011127242/category_10823035.html?spm=1001.2014.3001.5482
- 1

```
    deployment-clusterrole   ClusterRole
       Deployment Statefulset Daemonset
     app-team1        cicd-token   ServiceAccount
  ClusterRole   ServiceAccount          app-team1


kubectl create ns app-team1

kubectl create serviceaccount cicd-token -n app-team1

kubectl create clusterrole deployment-clusterrole --verb=create --resource=deployment,statefulset,daemonset

 kubectl -n app-team1 create rolebinding cicd-clusterrole   --clusterrole=deployment-clusterrole --serviceaccount=app-team1:cicd-token
```

- 2

```
  ek8s-node-1
           pod

kubectl cordon ek8s-node-1
kubectl drain ek8s-node-1 --ignore-daemonsets --delete-local-data --force
```

- 3

```
  master    1.20.1
    drain master
   worker node     manager  etcd  CNI   DNS

https://kubernetes.io/zh/docs/tasks/administer-cluster/kubeadm/kubeadm-upgrade/
kubectl get nodes
ssh mk8s-master-0
kubectl cordon mk8s-master-0
kubectl drain mk8s-master-0 --ignore-daemonsets
apt-mark unhold kubeadm kubectl kubelet
apt-get update && apt-get install -y kubeadm=1.20.1-00 kubelet=1.20.1-00 kubectl=1.20.1-00
apt-mark hold kubeadm kubectl kubelet
kubeadm upgrade plan
kubeadm upgrade apply v1.20.1 --etcd-upgrade=false
// kubectl rollout undo deployment coredns -n kube-system ,      rollout coredns          rollover
kubectl uncordon mk8s-master-0
```

- 4

```
  https://127.0.0.1:2379    etcd     /var/lib/backup/etcd-snapshot.db
       /data/backup/etcd-snapshot-previous.db    etcd
     ca.crt    etcd-client.crt  etcd-client.key


  etc,
ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379    --cacert=/etc/kubernetes/pki/etcd/ca.crt  --cert=/etc/kubernetes/pki/etcd/peer.crt   --key=/etc/
kubernetes/pki/etcd/peer.key  snapshot save   /var/lib/bacp/etcd-snapshot.db

  etcd
ETCDCTL_API=3 etcdctl --write-out=table snapshot   status   etcd-snapshot.db   --cacert=/etc/kubernetes/pki/etcd/ca.crt  --cert=/etc/kubernetes/pki/etcd/
peer.crt   --key=/etc/kubernetes/pki/etcd/peer.key


  ectd
ETCDCTL_API=3 etcdctl --write-out=table snapshot   restore    etcd-snapshot.db   --cacert=/etc/kubernetes/pki/etcd/ca.crt  --cert=/etc/kubernetes/pki/etcd/
peer.crt   --key=/etc/kubernetes/pki/etcd/peer.key
```

- 5

```
  services-networking/network-policies
         ns   fubar     80
  namespaceSelector   ns my-app  labels

https://kubernetes.io/docs/concepts/services-networking/network-policies/
[root@k8s-node1 ~]# vim NetworkPolicy.yaml

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-port-from-namespace
  namespace: fubar
spec:
  podSelector:
    matchLabels: {}
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          my-app-key: my-app-value
    - podSelector:
        matchLabels: {}
    ports:
    - protocol: TCP
      port: 80
```

- 6 < >

```
        deployment front-end         http       80/TCP
      front-end-svc   service       http
    service     NodePort

1 edit front-end   containers
kubectl edit deployment front-end
ports:
- name: http
  protocol: TCP
  containerPort: 80

https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#expose    kubectl expose deployment      -h
2 [root@k8s-node1 ~]# kubectl expose deployment  kual00612  --port=80 --target-port=80   --type=NodePort   --name=front-end-svc
```

- 7

```
       Ingress       ping      ing-internal
   /hello        hello   5678

https://kubernetes.io/docs/concepts/services-networking/ingress/

[root@k8s-node1 ~]# vim ingress.yaml

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ping
  namespace: ing-internal
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /hello
        pathType: Prefix
        backend:
          service:
            name: test
            port:
              number: 5678
```

- 8

```
   deployment guestbook   6 pod

kubectl scale deployment --replicas=6 guestbook

kubectl edit  deployment guestbook  #  replicas    6
```

- 9

```
   pod    nginx-kusc0041    nginx
    pod   disk=ssd
https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/

apiVersion: v1
kind: Pod
metadata:
  name: nginx-kusc0041
spec:
  containers:
  - name: nginx
    image: nginx
  nodeSelector:
    disk: ssd
```

- 10

```
               nodes tainted Noschedule        /opt/kusco0402/kusco0402.txt
[root@k8s-node1 ~]# kubectl get nodes|grep -v NAME|wc -l
2
[root@k8s-node1 ~]# kubectl describe nodes |grep NoSchedule|wc -l
0
```

```
pods   NoSchedule
echo 2 > /opt/kusco0402/kusco0402.txt
```

- 11

```
      kucc1   pod
pod     nginx   redis

[root@k8s-node1 ~]# more create-pods.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-to-redis
spec:
  containers:
  - name: redis
    image: redis
  - name: nginx
    image: nginx
```

- 12

```
       app-config  PV PV    2Gi       ReadWriteMany volume      hostPath

pv     hostPath   /srv/app-config

                      hostPath   /srv/app-config
https://kubernetes.io/zh/docs/tasks/configure-pod-container/configure-persistent-volume-storage/#create-a-persistentvolume
[root@k8s-node1 ~]# more create-pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: app-config-1
  labels:
    type: local
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/srv/app-config"
```

- 13

```
     storageclass csi-hostpath-sc         pv-volume   pvc     10Mi
      web-server  pod   nginx     /usr/share/nginx/html       pvc
    pvc      10Mi     70Mi
https://kubernetes.io/docs/concepts/storage/persistent-volumes/
https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/
[root@k8s-node1 ~]# cat  create-pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pv-volume
spec:
  storageClassName: csi-hostpath-sc
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Mi
[root@k8s-node1 ~]# cat  pvc-mount-pods.yaml
apiVersion: v1
kind: Pod
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: my-pvc
      persistentVolumeClaim:
        claimName: pv-volume
  containers:
    - name: web-server
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: my-pvc
kubectl edit pvc pv-volume --record   #--record
```

- 15

```
   sidecar
     sidecar   (   busybox    )     pod 11-factor-app
   sidecar        /var/log/11-factor-app.log
   volume    /var/log       sidecar     11-factor-app.log

   kubectl get pod -o yaml         pod        pod 11-factor-app
copy     yaml             sidecar
   emptyDir              /var/log
     sidecar   pod     kubectl logs
https://kubernetes.io/zh/docs/concepts/cluster-administration/logging/
```

- 16

```
   wk8s-node-0     NotReady             Ready


       get nodes               kubelet      status
   kubelet   enable kubelet

kubectl get nodes
ssh wk8s-node-0
sudo -i
systemctl status kubelet
systemctl enable kubelet
systemctl restart kubelet
systemctl status kubelet

   get nodes         Ready
```

- 17

```
   label   name=cpu-loader   pod,    cpu         pod           /opt/KUTR00401/KUTR00401.txt

      top      -l label_key=label_value   --sort=cpu
kubectl top pod -l name=cpu-loader -A --sort-by=cpu
echo podName >> /opt/KUTR00401/KUTR00401.txt
```

- sidecar

```
     sidecar   (  busybox   )     pod 11-factor-app
   sidecar        /var/log/11-factor-app.log
   volume    /var/log       sidecar     11-factor-app.log

     https://kubernetes.io/zh/docs/concepts/cluster-administration/logging/
  kubectl get podname -o yaml > podname.yaml    yaml          pod
    copy      yaml    sidecar               emtpyDir   /var/log       apply     sidecar   pod
pod      ,   kubectl logs 11-factor-app sidecar

#
[root@k8s-node1 ~]# kubectl  get pods  11-factor-app -o yaml >    11-factor-app.ayml
#

[root@k8s-node1 ~]# more sidecar.yaml
apiVersion: v1
kind: Pod
metadata:
  name: 11-factor-app
spec:
  containers:
  - name: 11-factor-app
    image: busybox
    args:
    - /bin/sh
    - -c
    - >
      i=0;
      while true;
      do
        echo "$(date) INFO $i" >> /var/log/11-factor-app.log;
        i=$((i+1));
        sleep 1;
      done
    volumeMounts:
    - name: varlog
      mountPath: /var/log
  - name: sidecar
    image: busybox
    args: [/bin/sh, -c, 'tail -n+1 -f /var/log/11-factor-app.log']
    volumeMounts:
    - name: varlog
      mountPath: /var/log
  volumes:
```

```
  - name: varlog
    emptyDir: {}
```

- PVC

```
     storageclass csi-hostpath-sc         pv-volume   pvc     10Mi
      web-server  pod   nginx     /usr/share/nginx/html       pvc
    pvc      10Mi     70Mi

         k8s        StorageClass
[root@k8s-node1 newpvc]# more createpvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pv-volume-to2
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 10Mi
  storageClassName: cis-hostpath-cs-to2

[root@k8s-node1 newpvc]# kubectl  get pvc #
NAME            STATUS    VOLUME               CAPACITY   ACCESS MODES   STORAGECLASS          AGE
pv-volume-to2   Bound     task-pv-volume-to2   1Gi        RWO            cis-hostpath-cs-to2   31m

[root@k8s-node1 newpvc]# more newpvcmount.yaml
apiVersion: v1
kind: Pod
metadata:
  name: web-server-to2
spec:
  containers:
    - name: nginx
      image: nginx
      volumeMounts:
      - mountPath: "/usr/share/nginx/html"
        name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: pv-volume-to2

 kubectl  edit   pvc pv-volume-to2   --record  #  nfs
```

- Ingress

```
       Ingress       ping      ing-internal
     /hello        hello   5678     #    ingress      ip
```

CKA K8s

-
- txt html,
- https://www.jianshu.com/p/a743860b13fe

- https://www.cloudcared.cn/3138.html

: 2022-05-08