1. 监控 foobar Pod 的日志，提取 pod 相应的行'error'写入到/logs 文件中

```
//XXX表示pod名称
kubectl logs XXX

//grep后面可以加上搜索的关键字
kubectl logs|grep YYY

列出指定pod的日志中状态为Error的行，并记录在指定的文件上

kubectl logs <podname> | grep error > /opt/KUCC000xxx/KUCC000xxx.txt
```

2. 使用 capacity 排序列出所有的 PV，把输出内容存储到/opt/xxx.file文件中

利用第13题创建PV，看好**namespace**，通过这个命令来查看字段的级别

```
$ kubectl get pv/pv0003 --all-namespaces -o yaml
```

解题：

```
  $ kubectl get pv --all-namespaces --sort-by=.spec.capacity.storage >
/opt/xxx.file
```

3. 确保在集群的每个节点上运行一个 Nginx Pod。其中 Nginx Pod 必须使用 Nginx 镜像。不要覆盖当前环境中的任何 traints。 使用 Daemonset 来完成这个任务，Daemonset 的名字使用 ds。

- 文档位置：https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/#writing-a-daemonset-spec

- 思路：删除tolerations字段，复制到image: gcr.io/fluentd-elasticsearch/fluentd:v2.5.1这里即可，再按题意更改yaml文件。

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: ds
spec:
  # 如果需要覆盖master的taints，就需要把下面这三行加上
  # tolerations:
  # - key: node-role.kubernetes.io/master
  #   effect: NoSchedule
  selector:
    matchLabels:
      name: nginx
  template:
    metadata:
      labels:
        name: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
```

4. initcontainer，这个 initcontainer 应该创建一个名为/workdir/calm.txt 的空文件，如果/workdir/calm.txt 没有被检测到，这个 Pod 应该退出

- 思路：题目中yaml文件已经给出，只需要增加initcontainers部分，以及emptyDir: {} 即可
- 文档位置：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/#define-a-liveness-command

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: init-container
spec:
  containers:
  - name: nginx
```

```
    image: nginx
    volumeMounts:
    - mountPath: /workdir
      name: workdir
    livenessProbe:
      exec:
        command:
        - cat
        - /workdir/calm.txt
  initContainers:
  - name: init-myservice
    image: busybox
    command: ['sh', '-c', "touch /workdir/calm.txt"]
    volumeMounts:
    - mountPath: /workdir
      name: workdir
  volumes:
  - name: workdir
    emptyDir: {}
```

5. 创建一个名为 kucc 的 Pod,其中内部运行着 nginx+redis+memcached+consul 4 个容器

- 文档位置：https://kubernetes.io/docs/concepts/workloads/pods/init-containers/#using-init-containers
- 思路：考试时候有可能随机挑选其中两个容器

```
apiVersion: v1
kind: Pod
metadata:
  name: kucc
spec:
  containers:
  - name: nginx
    image: nginx
  - name: redis
    image: redis
  - name: memcached
    image: memcached
  - name: consul
    image: consul
```

6. 创建 Pod，名字为 nginx，镜像为 nginx，添加 labels env=test，nodeSelector disk=ssd

- 文档位置：https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#nodeselector
- 两个要求，一个是给pod添加label，一个是nodeSelector，不要忘记

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
    nodeSelector:
      disk: ssd
```

7. 创建 deployment 名字为 nginx-app 容器采用 1.11.9 版本的 nginx 这个 deployment 包含 3 个副本,接下来通过滚动升级的方式更新镜像版本为 1.12.0，并记录这个更新，最后，回滚这个更新到之前的 1.11.9 版本

- 文档位置：https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#creating-a-deployment
- 思路：创建deployment时候，可以使用命令或者配置文件，使用命令的好处是快，如果命令忘了，就查文档，用yaml文件
- 可以使用命令创建，但是需要使用`kubectl edit`命令去动态修改副本数

```
$ kubectl create deployment nginx-app --image=nginx:1.11.9
$ kubectl edit deployment/nginx-app
```

- 建议大家使用yaml的方式创建deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.11.9
        ports:
        - containerPort: 80
```

```
$ kubectl set image deployment/nginx-app nginx=nginx:1.12.0 --record
(nginx-app container名字)
$ kubectl rollout undo deployment nginx-app
## 查看历史记录
$ kubectl rollout history deployment nginx-app
## 回滚到指定的历史版本
$ kubectl rollout history deployment.v1.apps/nginx-app --revision=2
```

8. 创建和配置 service，名字为 front-end-service。可以通过 NodePort/ClusterIp 开访问，并且路由到 front-end 的 Pod 上

- 文档位置：https://kubernetes.io/docs/tutorials/services/#source-ip-for-services-with-type-nodeport
- 思路：可以使用命令或者配置文件，建议使用命令，比较快

```
$ kubectl expose pod fron-end --name=front-end-service --port=80  --
type=NodePort
$ kubectl expose deployment source-ip-app --name=nodeport --port=80 --target-
port=8080 --type=NodePort
```

9. 创建一个 Pod，名字为 Jenkins，镜像使用 Jenkins。在新的 namespace website-frontend 上创建

- 文档位置：https://kubernetes.io/docs/concepts/workloads/pods/init-containers/#using-init-containers
- 思路：先创建namespace，再创建pod

```
$ kubectl create ns website-frontend
```

```
apiVersion: v1
kind: Pod
metadata:
  name: Jenkins
  namespace: website-frontend
spec:
  containers:
  - name: Jenkins
    image: Jenkins
```

10. 创建 deployment 的 spec 文件: 使用 redis 镜像，7 个副本，label 为 app_enb_stage=dev deployment 名字为 kual00201 保存这个 spec 文件到/opt/KUAL00201/deploy_spec.yaml完成后，清理(删除)在此任务期间生成的任何新的 k8s API 对象

- 文档位置：https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#creatin

g-a-deployment

- 思路：1.18版本中不能指定replica，最好使用yaml文件

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kual00201
  labels:
    app_enb_stage: dev
spec:
  replicas: 7
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
    spec:
      containers:
      - name: redis
        image: redis
```

- 也可以使用命令创建，但是一定要注意，有些选项有可能是不可用的

```
$ kubectl create deployment kual00201 --image=redis --dry-run -oyaml >
q10.txt
$ vim q10.txt
```

11. 列某个namepsace 下某个service所代表的所有pod的名，这次是environment=production

```
//第一步，找到某个service下的labels
kubectl get service --show-labels

//第二步，可以按照po标签，根据-l标签来进行筛选
kubectl get pods -l environment=production

//第三步，直接找到某个节点名，进行输出
kubectl get pods -l environment=production -o=custom-
columns=NAME:metadata.name >name.yaml
```

也可以通过svc的selector去找

12. 创建一个secret,名字为super-secret包含username: bob,创建pod1挂载该secret，路径为/secret，创建pod2，使用环境变量引用该secret，该变量的环境变量名为ABC

文档位置：https://kubernetes.io/docs/concepts/configuration/secret/#creating-a-secret-manually

```
$ echo -n "bob" | base64
Ym9i
```

创建secret

```
apiVersion: v1
kind: Secret
metadata:
  name: super-secret
type: Opaque
data:
  username: Ym9i
```

把secret当做文件使用，https://kubernetes.io/docs/concepts/configuration/secret/#using-secrets

```
apiVersion: v1
kind: Pod
metadata:
  name: pod1
spec:
  containers:
  - name: mypod
    image: nginx
    volumeMounts:
    - name: foo
      mountPath: "/secret"
      readOnly: true
  volumes:
  - name: foo
    secret:
      secretName: super-secret
```

把secret当做环境变量，https://kubernetes.io/docs/concepts/configuration/secret/#using-secrets-as-environment-variables

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod2
spec:
  containers:
  - name: nginx
    image: nginx
    env:
      - name: ABC
        valueFrom:
          secretKeyRef:
            name: super-secret
            key: username
  restartPolicy: Never
```

13. 在新的ns中创建pv，指定pv名字和挂载路径，等

- 文档位置： https://kubernetes.io/docs/tasks/configure-pod-container/configure-persistent-volume-storage/#create-a-persistentvolume

```
kubectl create ns new
```

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0001
spec:
  storageClassName: manual
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/etc/data"
```

14. 为给定deploy websity副本扩容到6

```
$ kubectl scale deployment website --replicas=6
```

15. 查看给定集群ready的node个数(不包含NoSchedule)

```
# 查看所有的nodes，吧Ready的节点挑出来
kubectl get nodes
# 查看污点，taints，把所有ready得都执行，如果有NoSchedule那就不算数
kubectl describe node $nodename | grep Taint
```

16. 找出指定ns中使用cup最高的pod名写出到指定文件

安装metric-server，下载清单文件，git地址是https://github.com/kubernetes-sigs/metrics-server

```
wget https://github.com/kubernetes-sigs/metrics-
server/releases/download/v0.3.6/components.yaml
```

编辑components文件，在这个配置下面添加配置

```
        args:
          - --cert-dir=/tmp
          - --secure-port=4443
          # 添加下面这个
          - --kubelet-insecure-tls
```

使用下面的命令

```
$ kubectl top pod --sort-by=cpu --namespace=default
$ kubectl top pod --sort-by=cpu --namespace=default --no-headers|head -1|awk
'{print $1}' > xx.file
```

17. 创建一个 deployment 名字为:nginx-dns 暴露的server名为：nginx-dns 确保service和 pod 可以通过各自的 DNS 记录访问 容器使用 nginx 镜像，使用 nslookup 工具来解析 service 和 pod 的记录并写入相应的/opt/service.dns 和/opt/pod.dns 文件中，确保你使用 busybox:1.28 的镜像用来测试。

deployment和svc都是创建好的，我们只需要创建busybox并且解析就好了。

创建deployment和svc

```
$ kubectl create deployment nginx-dns --image=nginx
$ kubectl expose deployment nginx-dns --name=nginx-dns --port=80 --
type=NodePort
```

建立busybox，https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/#pods

```
apiVersion: v1
kind: Pod
metadata:
  name: busybox
spec:
  containers:
  - image: busybox
    command:
      - sleep
      - "3600"
    name: busybox
```

解析

```
$ kubectl exec -it busybox -- nslookup nginx-dns > /opt/service.dns
# POD的IP地址是通过kubectl get pods -o wide查看的，这里是10.244.0.122
$ kubectl exec -it busybox -- nslookup 10.244.0.122(pod IP) > /opt/pod.dns
```

18. 给定https地址，ca，cert证书，key备份该数据到指定目录

文档位置：https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/#backing-up-an-etcd-cluster

```
$ ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 --
cacert=/pki/ca.crt --cert=/pki/cert.crt --key=/pki/key.crt snapshot save 给的路径
# 有些题目下--ca-file会报错，记得看etcdctl -h里的字段怎么要求的，端口可以使用netstat -untlp
$ ETCDCTL_API=3
/var/lib/docker/overlay2/c3d4d1cd709732d880752e384677513ce66906e266c6f6607af0ea20452ac21d/merged/usr/local/bin/etcdctl --endpoints=https://127.0.0.1:2379 --
cacert=/etc/kubernetes/pki/etcd/ca.crt --
cert=/etc/kubernetes/pki/etcd/server.crt --
key=/etc/kubernetes/pki/etcd/server.key snapshot save /tmp/1
```

19. 在ek8s集群中使name=ek8s-node-1节点不能被调度，并使已被调度的pod重新调度

```
$ kubectl drain node1 --ignore-daemonsets --delete-local-data
```

20. 给定集群中的一个node未处于ready状态，解决该问题并具有持久性

```
进入集群
ssh node

systemctl status kubelet # 应该是没启动，或者宕了

systemctl start kubelet
systemctl enable kubelet
```

21. 题目很绕，大致是 在k8s的集群中的node1节点配置kubelet的service服务，去拉起一个由kubelet直接管理的pod(说明了是静态pod),

这个是在node01上的/etc/kubernetes/manifest 新建了静态pod的yaml，但回到master仍然static pod没启动，也是在node01上的kubelet.service里少了static pod的配置，kubelet.service中的/var/lib/kubelet/config.yaml文件中添加配置staticPodPath: /etc/kubernetes/manifest 然后再重启kubelet.service，再会master上看就起来了

还有一种可能

他kubelet.service文件里的static pod配置瞎给了个乱码，把static-pod-path 改成正确的/etc/kubernetes/manifest ，然后重启kubelet.service 好像就好了

22. 某集群中kubelet.service服务无法正常启动，解决该问题，并具有持久性

```
# 这个集群是使用二进制方式安装的
# kubectl 命令能用 kubectl get cs 健康检查  看manager-controller  是否ready   如果不ready
$ systemctl start kube-manager-controller.service
# 使用下面的命令来校验集群是否正常
$ kubectl get nodes
```

23.创建带hostpath的pod

文档位置：https://kubernetes.io/docs/concepts/storage/volumes/#hostpath

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: nginx
    name: nginx
```

```
    volumeMounts:
    - mountPath: /test-pd
      name: test-volume
  volumes:
  - name: test-volume
    hostPath:
      path: /etc
```

24. 使用kubeadm初始化集群，集群中有两个节点，一个master，一个node，只要kubectl get nodes命令看到两个节点处于ready状态就算通过。kubeadm init时要用他给的一个配置文件，也就是kubeadm init --config=/etc/xxx.config --ignorexxxx 初始化的

- 文档位置：https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

如果是没安装好的，他会提示你怎样安装

```
$ apt install kubeadm kubectl kubelet
```

如果是安装好的，他会告诉你，已经安装完成

```
$ kubectl
$ kubeadm -h
$ systemctl status kubelet
```

考试时他会提示你加上配置--ignorexxx, 不然init 或者join cluster时会报错，进行不下去，如果使用calico作为网络的方案，一定要加上`--pod-network-cidr=192.168.0.0/16`，输入命令的时候，记得切换到root用户，或者使用`sudo`，下面的命令在master节点上运行。

```
kubeadm init --config=/etc/xxx.config --ignore-preflight-errors xxx --pod-network-cidr=192.168.0.0/16
```

出现这个提示就算成功

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

```
You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each
as root:

kubeadm join 10.0.1.50:6443 --token x6n83i.30xvcgznrubhh9st \
    --discovery-token-ca-cert-hash
sha256:c5fd69afbd7f725663b26133d4e0f1a07674e7058dcb7424de77f8390a7ca7df
```

执行命令，配置kubectl

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

再到node节点上执行

```
kubeadm join 10.0.1.50:6443 --token x6n83i.30xvcgznrubhh9st \
    --discovery-token-ca-cert-hash
sha256:c5fd69afbd7f725663b26133d4e0f1a07674e7058dcb7424de77f8390a7ca7df
```

安装完成之后，运行

```
kubectl apply -f https://docs.projectcalico.org/v3.11/manifests/calico.yaml
```

最后在master节点上执行

```
ubuntu@ip-10-0-1-50:~$ kubectl get nodes
NAME            STATUS   ROLES    AGE     VERSION
ip-10-0-1-189   Ready    <none>   89s     v1.18.2
ip-10-0-1-50    Ready    master   2m34s   v1.18.2
```

看到status是Ready就是正确的了