

# Kubernetes\_DaemonSet&Job

## - 실습가이드

최초 작성일 : 2024/03/04

최종 제출일 : 2024/03/05

김민경

### 목차

I. 개념정리-----	2
1. CoreDNS-----	2
2. Daemonset-----	3
3. 노드 스케줄링 - Taint & Toleration-----	4
4. Job & Cronjob-----	5
II. 실습-----	6
1. CoreDNS-----	6
2. Daemonset-----	7
3. Tolerations-----	9
4. Job-----	13
5. Cronjob-----	15

# I. 개념정리

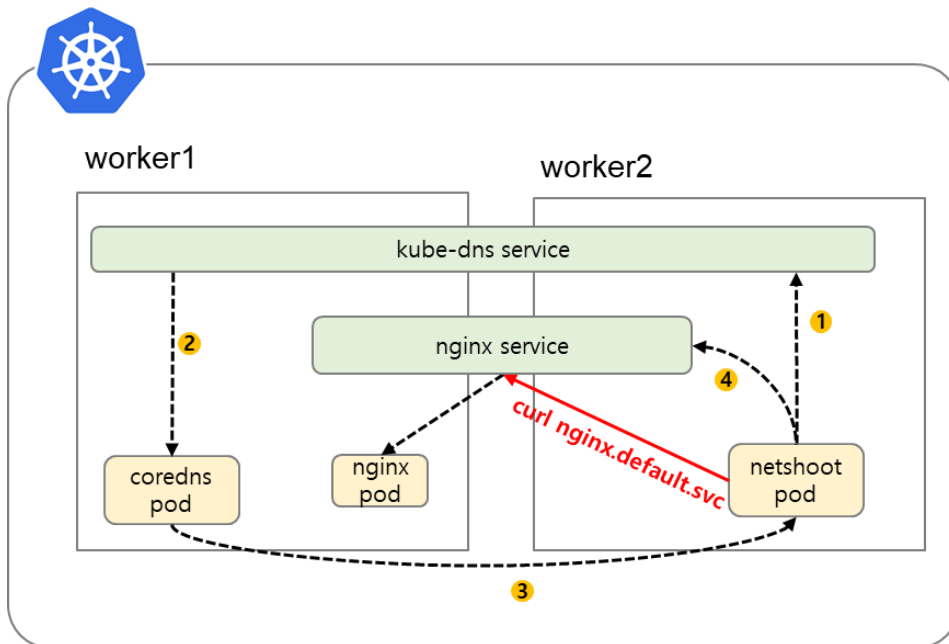
## 1. CoreDNS

### 1-1. 개념

- 쿠버네티스 클러스터의 DNS 역할을 수행할 수 있는, 유연하고 확장 가능한 DNS 서버
- 클러스터를 지속적으로 모니터링하며, 새로운 서비스 or Pod가 추가되는 경우, 도메인 서버에 이를 업데이트함

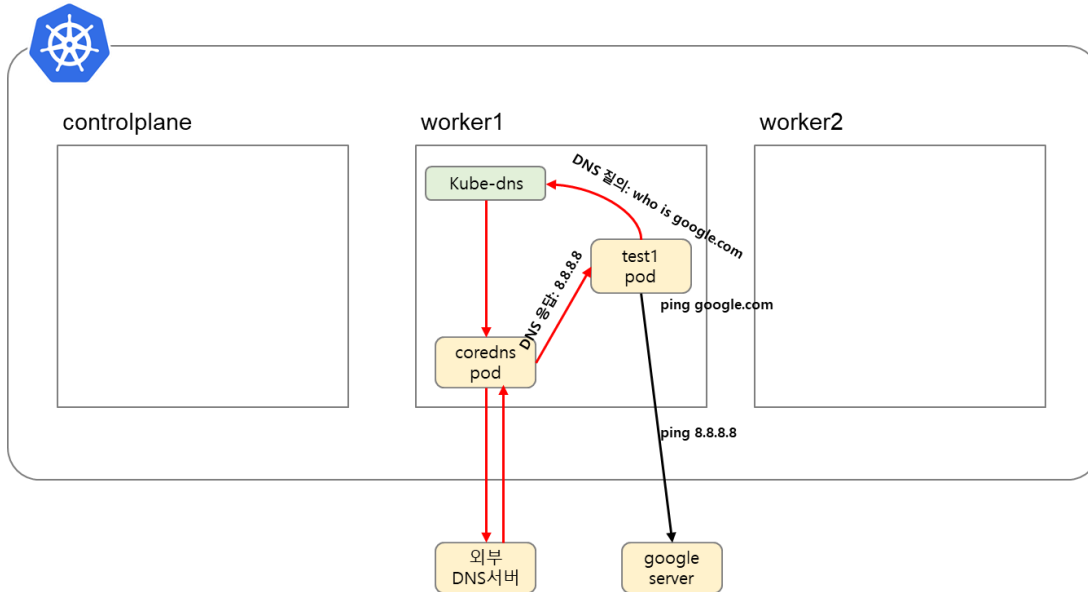
### 1-2. DNS 요청 과정

#### 1) 클러스터 내부 DNS 요청 과정



- ① 파드가 서비스 이름으로 dns 쿼리를 시작함
- ② 각 pod 는 디폴트로 coredns 를 바라보기 때문에 모든 dns 요청이 coredns 로 가게됨
- ③ coredns는 쿠버네티스의 서비스와 파드에 대한 DNS 정보를 관리하므로 내부 쿼리에 대한 정보를 가지고 있음
- ④ coreDNS는 해당 서비스의 클러스터 IP 주소로 요청을 해석하고 응답함
- ⑤ 파드는 받은 응답을 사용해 해당 서비스와 통신함

## 2) 클러스터 외부 DNS 요청 과정

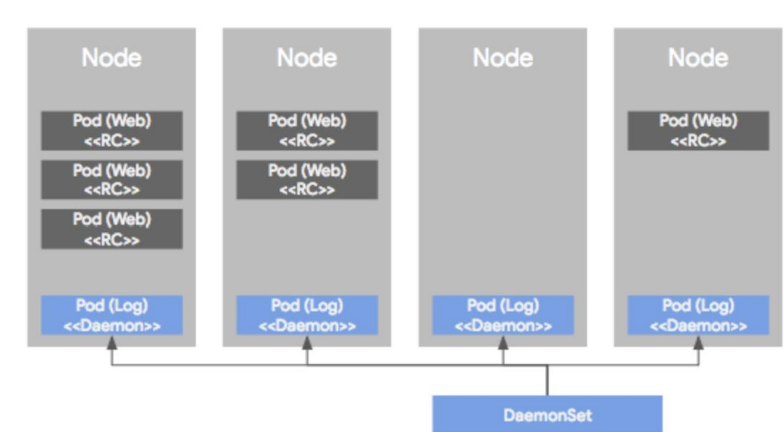


- 쿠버네티스 클러스터의 외부로 나가는 DNS 쿼리 또한 coredns 가 담당함
- 각 pod 는 디폴트로 coredns 를 바라보기 때문에 모든 dns 요청이 coredns 로 가게됨

- ① ping google.com 을 하게 되면
- ② coredns 가 DNS 질의를 받아 구글 서버 IP 주소를 구함
- ③ 그리고 DNS 질의한 pod 에게 구글 서버 ip 를 전달함
- ④ pod 는 전달받은 ip 를 이용해 구글 서버에 ping 을 수행함

## 2. Daemonset

### 2-1. 개념



- scheduler 와 무관하게 모든 노드에 pod 를 동일하게 하나씩 생성해주는 controller

#### 참고) k8s의 controller

- 기본 오브젝트를 생성하고 관리하는 역할
- Replica Set, DaemonSet, Job, StatefulSet, Deployment 등이 있음

- 모든 노드에 각 1 개의 pod 생성

- deployment 와 유사하게 파드를 생성 및 관리함

- deployment : rolling update, rolling back 등 배포 작업을 세분화

- daemonset : 특정 노드 or 모든 노드에 실행되어야할 특정 파드를 관리하는 것

- 주로 로그 수집기를 실행하거나 노드를 모니터링하는 등 클러스터 전체에 항상 실행시켜두어야 하는 파드를 실행할 때 사용

- taint & tolerant 옵션 사용 시 daemonset 을 전체 클러스터 노드가 x 닌 특정 노드들에만 선택해서 실행 가능 함

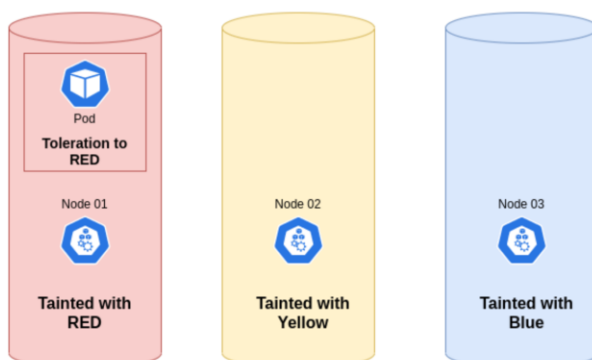
- taint 가 지정된 노드에서는 파드가 뜰 수 x 거나, 강제로 파드를 띄우기 위해 tolerant 옵션을 줄 수 있음

° tolerant 옵션은 taint 보다 더 우선순위가 높음

## 3. 노드 스케줄링 - Taint & Toleration

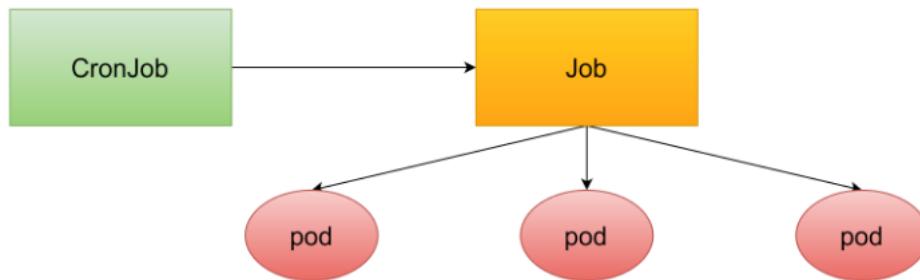
- 노드 스케줄링을 임의로 핸들링하기 위한 방법 중 하나

### 3-1. 개념



- **Taint** : 해당 노드에 파드를 제외하는 설정 (일종의 '자물쇠' 역할)
- **Toleration** : 해당 노드에 파드를 스케줄링 하는 설정 (자물쇠를 푸는 일종의 '열쇠' 역할)

## 4. Job & Cronjob



### 4-1. Job

- pod 와 달리 종료되는 것을 기대하고 작업 실행결과를 알려줌 (Completed 상태를 기대함)
  - pod 는 시작되면 항상 실행되는 것을 기대함 (running 상태를 기대함)
- 쿠버네티스의 controller 중 하나인 job controller 가 이를 관리함
- pod 를 이용해 일회성 or 정기적인 작업을 실행할 때 사용함
  - 주로 백업, 배포 전의 check 를 위해(버전 확인용도), 재난 문자(특정 이벤트 발생 시 그 때 한 번 발생하고 메시지 돌리고 끝냄, 필요시 마다 적은 리소스로 동작 가능)에 사용

### 4-2. Cronjob

- 주기적으로 특정 동작을 수행하고 종료하는 작업(배치 작업)을 정의하기 위한 리소스
  - 정해진 스케줄에 따라 Job 을 실행하는 Kubernetes 리소스
- 내부적으로 job 을 생성해 작업을 수행함
  - job 은 내부적으로 pod 를 만듦
- 일회성 작업을 주기적으로 반복하는 자동화 작업에 유용함
  - 주기적으로 데이터 백업하거나 데이터 점검 및 알림 전송 등의 목적으로 사용

## II. 실습

### 1. Coredns

- Coredns 개수는 deployment로 조절 가능함

#### 1-1. Deployment 목록 확인하기

- `k get deployments.apps -A`

//쿠버네티스 클러스터 전체에서 실행 중인 모든 Deployment의 목록과 그 상태를 보여주며, 네임스페이스별로 구분하여 출력

```
(calico:N/A) root@k8s-m:~# k get deployments.apps -A
```

NAMESPACE	NAME	READY	UP-TO-DATE	AVAILABLE	AGE
calico-apiserver	calico-apiserver	2/2	2	2	2d2h
calico-system	calico-kube-controllers	1/1	1	1	2d2h
calico-system	calico-typha	2/2	2	2	2d2h
default	deployment-nginx	3/3	3	3	18h
kube-system	coredns	2/2	2	2	2d2h
kube-system	metrics-server	1/1	1	1	2d2h
local-path-storage	local-path-provisioner	1/1	1	1	2d2h
tigera-operator	tigera-operator	1/1	1	1	2d2h

```
(calico:N/A) root@k8s-m:~#
```

- kube-system과 관련된 deployment만 확인하기

```
k get deployments.apps -n kube-system
```

```
(calico:N/A) root@k8s-m:~# k get deployments.apps -n kube-system
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
coredns	2/2	2	2	2d2h
metrics-server	1/1	1	1	2d2h

- coredns가 현재 2개 돌고 있음ㄱ

⇒ deployment로 coredns 조절가능!

#### 1-2. coredns 개수 조정하기

```
k scale deployment --replicas 1 coredns -n kube-system && kubectl get pod -n kube-system -w
```

// coredns 개수 1개로 조정하고, kube-system 네임스페이스에 있는 모든 파드들의 상태를 실시간으로 보여주기

```
(calico:N/A) root@k8s-m:~# k scale deployment --replicas 1 coredns -n kube-system && kubectl get pod -n kube-system -w
deployment.apps/coredns scaled
NAME                                READY   STATUS    RESTARTS   AGE
coredns-5d78c9869d-vxmvm           1/1     Running   6 (30m ago) 2d2h
coredns-5d78c9869d-zjpp4           1/1     Terminating 6 (30m ago) 2d2h
etcd-k8s-m                          1/1     Running   6 (30m ago) 2d2h
kube-apiserver-k8s-m               1/1     Running   6 (30m ago) 2d2h
kube-controller-manager-k8s-m      1/1     Running   14 (4m41s ago) 2d2h
kube-proxy-9grng                   1/1     Running   6 (30m ago) 2d2h
kube-proxy-hsj8d                   1/1     Running   4 (27m ago) 2d2h
kube-proxy-zthkt                   1/1     Running   5 (29m ago) 2d2h
kube-scheduler-k8s-m               1/1     Running   13 (4m42s ago) 2d2h
metrics-server-64679cff5c-8b4n9    1/1     Running   12 (28m ago) 27h
coredns-5d78c9869d-zjpp4           1/1     Terminating 6 (30m ago) 2d2h
coredns-5d78c9869d-zjpp4           0/1     Terminating 6 2d2h
coredns-5d78c9869d-zjpp4           0/1     Terminating 6 2d2h
coredns-5d78c9869d-zjpp4           0/1     Terminating 6 2d2h
coredns-5d78c9869d-zjpp4           0/1     Terminating 6 2d2h
```

- 한 coredns가 1개가 terminating 되는 것을 확인

- coredns 개수 한 개로 줄어든 것 확인하기

```
^C(calico:N/A) root@k8s-m:~# get deployments.apps -n kube-system-w
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
coredns         1/1     1             1           2d2h
metrics-server  1/1     1             1           2d2h
```

## 2. Daemonset

### 2-1. daemonset 목록 확인하기

**k get daemonsets.apps -A**

```
(calico:N/A) root@k8s-m:~# k get daemonsets.apps -A
NAMESPACE   NAME           DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
calico-system  calico-node    3         3         3       3             3           kubernetes.io/os=linux 2d2h
calico-system  csi-node-driver 3         3         3       3             3           kubernetes.io/os=linux 2d2h
kube-system   kube-proxy     3         3         3       3             3           kubernetes.io/os=linux 2d2h
```

**k get daemonsets.apps -n kube-system**

```
(calico:N/A) root@k8s-m:~# k get daemonsets.apps -n kube-system
NAME           DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
kube-proxy     3         3         3       3             3           kubernetes.io/os=linux 2d2h
(calico:N/A) root@k8s-m:~#
```

⇒ 정리) Deployment으로 CoreDNS가, Daemonset으로 proxy가 관리되고 있음!

### 2-2. 상세정보 확인하기

**k describe daemonsets.apps -n kube-system kube-proxy**

- 아래 사진처럼 kube-proxy가 정상적으로 동작 중임

```
(calico:N/A) root@k8s-m:~# k describe daemonsets.apps -n kube-system kube-proxy
Name: kube-proxy
Selector: k8s-app=kube-proxy
Node-Selector: kubernetes.io/os=linux
Labels: k8s-app=kube-proxy
Annotations: deprecated.daemonset.template.generation: 1
Desired Number of Nodes Scheduled: 3
Current Number of Nodes Scheduled: 3
Number of Nodes Scheduled with Up-to-date Pods: 3
Number of Nodes Scheduled with Available Pods: 3
Number of Nodes Misscheduled: 0
Pods Status: 3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels: k8s-app=kube-proxy
  Service Account: kube-proxy
  Containers:
    kube-proxy:
      Image: registry.k8s.io/kube-proxy:v1.27.11
      Port: <none>
      Host Port: <none>
      Command:
        /usr/local/bin/kube-proxy
        --config=/var/lib/kube-proxy/config.conf
        --hostname-override=$(NODE_NAME)
      Environment:
        NODE_NAME: (v1:spec.nodeName)
      Mounts:
        /lib/modules from lib-modules (ro)
        /run/xtables.lock from xtables-lock (rw)
        /var/lib/kube-proxy from kube-proxy (rw)
  Volumes:
    kube-proxy:
      Type: ConfigMap (a volume populated by a ConfigMap)
      Name: kube-proxy
      Optional: false
    xtables-lock:
      Type: HostPath (bare host directory volume)
      Path: /run/xtables.lock
      HostPathType: FileOrCreate
    lib-modules:
      Type: HostPath (bare host directory volume)
      Path: /lib/modules
      HostPathType:
    Priority Class Name: system-node-critical
Events: <none>
(calico:N/A) root@k8s-m:~#
```

## k describe daemonsets.apps -n calico-system calico-node

// calico-system 네임스페이스 내의 calico-node 라는 DaemonSet 에 대한 상세 정보 표시하기

```
(calico:N/A) root@k8s-m:~# k describe daemonsets.apps -n calico-system calico-node
Name: calico-node
Selector: k8s-app=calico-node
Node-Selector: kubernetes.io/os=linux
Labels: <none>
Annotations: deprecated.daemonset.template.generation: 1
Desired Number of Nodes Scheduled: 3
Current Number of Nodes Scheduled: 3
Number of Nodes Scheduled with Up-to-date Pods: 3
Number of Nodes Scheduled with Available Pods: 3
Number of Nodes Misscheduled: 0
Pods Status: 3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels: app.kubernetes.io/name=calico-node
          k8s-app=calico-node
  Annotations: hash.operator.tigera.io/cni-config: 7c3d4f43aed6c9f83376215a5e9a368bef0902a0
                hash.operator.tigera.io/system: fdd45054a8ae4f629960ce37570929502e59449
                tigera-operator.hash.operator.tigera.io/tigera-ca-private: 375d14b5fa49ba409e8c48256bac66853872a7da
  Service Account: calico-node
  Init Containers:
    flexvol-driver:
      Image: docker.io/calico/pod2daemon-flexvol:v3.27.2
      Port: <none>
      Host Port: <none>
      SeccompProfile: RuntimeDefault
      Environment: <none>
      Mounts:
        /host/driver from flexvol-driver-host (rw)
  install-cni:
      Image: docker.io/calico/cni:v3.27.2
      Port: <none>
      Host Port: <none>
      SeccompProfile: RuntimeDefault
      Command:
        /opt/cni/bin/install
      Environment:
        CNI_CONF_NAME: 10-calico.conflist
        SLEEP: false
        CNI_NET_DIR: /etc/cni/net.d
        CNI_NETWORK_CONFIG: <set to the key 'config' of config map 'cni-config'> Optional: false
        KUBERNETES_SERVICE_HOST: 10.96.0.1
        KUBERNETES_SERVICE_PORT: 443
      Mounts:
        /host/etc/cni/net.d from cni-net-dir (rw)
        /host/opt/cni/bin from cni-bin-dir (rw)
  Containers:
```



## 3. Tolerations

### 3-1. Tolerations 설정 x 한 것

1) vim 파일 만들어 daemonset-1.yaml 파일 아래와 같이 구성하기

```
# daemonset-1.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: daemonset-1
spec:
  selector:
    matchLabels:
      name: daemonset-cloud
  template:
    metadata:
      labels:
        name: daemonset-cloud
    spec:
      containers:
      - name: daemonset-pod
        image: nginx
```

2) **cat daemonset-1.yaml** 로 파일 내용 확인하기

3) 모니터링 탭 만들기

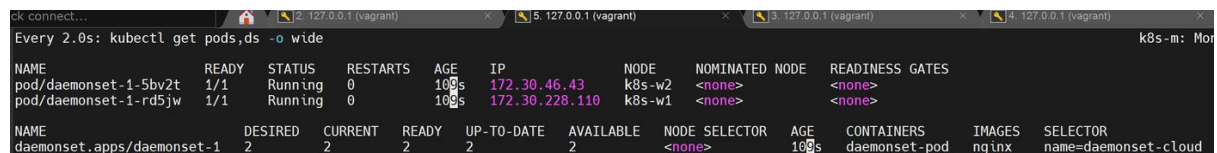
```
watch -d 'kubectl get pods,ds -o wide'
```

//ds : daemonset

4) 적용하기

```
k apply -f daemonset-1.yaml
```

5) 모니터링으로 확인하기



NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED	NODE	READINESS	GATES
pod/daemonset-1-5bv2t	1/1	Running	0	10s	172.30.46.43	k8s-w2	<none>		<none>	
pod/daemonset-1-rd5jw	1/1	Running	0	10s	172.30.228.110	k8s-w1	<none>		<none>	

NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE	NODE	SELECTOR	AGE	CONTAINERS	IMAGES	SELECTOR
daemonset.apps/daemonset-1	2	2	2	2	2	<none>		10s	daemonset-pod	nginx	name=daemonset-cloud

- 현재 워커노드 2개(w1, w2)에 pod 가 1 개씩 생성됨을 확인할 수 있음

° 원래 daemonset 은 노드마다 pod 1 개씩 있어야해 master 노드까지 3 개가 생성되어야 함

- 다른 방식으로 확인하기

### k get pod --show-labels

```
(calico:N/A) root@k8s-m:~# k get pod --show-labels
NAME          READY   STATUS    RESTARTS   AGE   LABELS
daemonset-1-5bv2t  1/1     Running   0          16m   controller-revision-hash=69c455fb6c,name=daemonset-cloud,pod-template-generation=1
daemonset-1-rd5jw  1/1     Running   0          16m   controller-revision-hash=69c455fb6c,name=daemonset-cloud,pod-template-generation=1
```

### k describe ds daemonset-1

```
(calico:N/A) root@k8s-m:~# k describe ds daemonset-1
Name:          daemonset-1
Selector:      name=daemonset-cloud
Node-Selector: <none>
Labels:        <none>
Annotations:   deprecated.daemonset.template.generation: 1
Desired Number of Nodes Scheduled: 2
Current Number of Nodes Scheduled: 2
Number of Nodes Scheduled with Up-to-date Pods: 2
Number of Nodes Scheduled with Available Pods: 2
Number of Nodes Misscheduled: 0
Pods Status: 2 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  name=daemonset-cloud
  Containers:
    daemonset-pod:
      Image:      nginx
      Port:       <none>
      Host Port:  <none>
      Environment: <none>
      Mounts:     <none>
      Volumes:    <none>
Events:
  Type     Reason             Age   From               Message
  ----     ------             -
  Normal   SuccessfulCreate    17m   daemonset-controller Created pod: daemonset-1-rd5jw
  Normal   SuccessfulCreate    17m   daemonset-controller Created pod: daemonset-1-5bv2t
```

### 6) pod 삭제하기

#### k delete pod --all

```
(calico:N/A) root@k8s-m:~# k delete pod --all
pod "daemonset-1-5bv2t" deleted
pod "daemonset-1-rd5jw" deleted
```

- but pod 가 다시 생성됨

```
Every 2.0s: kubectl get pods,ds -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP            NODE   NOMINATED NODE   READINESS GATES
pod/daemonset-1-dd5ls  1/1     Running   0          45s   172.30.228.111 k8s-w1  <none>           <none>
pod/daemonset-1-l5tth  1/1     Running   0          45s   172.30.46.44   k8s-w2  <none>           <none>

NAME          DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE   CONTAINERS   IMAGES   SELECTOR
daemonset.apps/daemonset-1  2         2         2       2             2           <none>          18m   daemonset-pod  nginx   name=daemonset-cloud
```

### 7) daemonset 삭제를 통해 pod 삭제하기

#### k delete ds daemonset-1

- 데몬셋 지우면 pod 지워짐

```
ck connect...
Every 2.0s: kubectl get pods,ds -o wide
No resources found in default namespace.
```

### 3-2. woker node 한 개 더 추가하기

#### 1) 현재 워커노드 삭제하기

```
k delete node k8s-w1
```

```
admissioncontroller/admissioncontroller created
(calico:N/A) root@k8s-m:~# k delete node k8s-w1
node "k8s-w1" deleted
```

- w1 삭제 확인하기

```
(calico:N/A) root@k8s-m:~# k get nodes
NAME      STATUS    ROLES    AGE   VERSION
k8s-m     Ready     control-plane  2d15h  v1.27.11
k8s-w2    Ready     <none>      2d15h  v1.27.11
```

#### 2) 워커노드의 CNI 삭제하기

```
root@k8s-w1:~# rm -r /etc/cni/net.d/*
root@k8s-w1:~# rm -r /etc/kubernetes/*
root@k8s-w1:~# kubeadm reset
```

##### ① /etc/cni/net.d/ 삭제하기

- /etc/cni/net.d/ : CNI(Container Network Interface) 플러그인에 의해 사용되는 네트워크 구성 파일을 저장함
- 클러스터에서 노드를 제거한 다음 나중에 해당 노드를 다시 사용하려고 하면, 이전에 설정해놓았던 네트워크 구성과 충돌할 수 있음  
⇒ so. 이 디렉토리를 삭제하기

##### ② /etc/kubernetes/ 삭제하기

- /etc/kubernetes/ : 쿠버네티스의 클러스터 구성과 관련된 파일
  - kubelet, kubeadm 및 기타 쿠버네티스 컴포넌트의 구성 파일과 인증서 등이 포함
- 클러스터에서 노드를 제거한 다음 나중에 해당 노드를 다시 사용하려고 하면, 이전의 클러스터의 구성정보나 인증서와 충돌할 수 있음  
⇒ so. 이 디렉토리를 삭제하기

#### 3) 마스터 노드의 토큰 재생성 및 join 명령어 생성하기

- 현재 토큰 확인하기

```
sudo kubeadm token list
```

- 토큰은 유효기간이 존재함

- 클러스터 구축 후에 유효기간이 지나면 토큰이 사라질 수 있으니 확인하기

```
(calico:N/A) root@k8s-m:~# sudo kubeadm token list
TOKEN                                TTL    EXPIRES    USAGES          DESCRIPTION                                EXTRA GROUPS
123456.1234567890123456             <forever>    <never>    authentication,signing    The default bootstrap token generated by 'kubeadm init'.    system:bootstrappers:kubeadm:default-node-token
```

- 토큰 재생성 및 클러스터에 join 하기

- 안전함을 위해 다시 생성해주기

**sudo kubeadm token create --print-join-command**

// **--print-join-command** : 생성된 토큰을 사용해 클러스터에 조인하기

```
(calico:N/A) root@k8s-m:~# sudo kubeadm token create --print-join-command
kubeadm join 192.168.56.200:6443 --token vlybu0.e41soxtqv4muluua --discovery-token-ca-cert-hash sha256:44862b0c29b31870130ee6c994c9105bfae7ad553dff844e7929a6b291bf6ecf
```

- w1 과 연결해주기

```
root@k8s-w1:~# kubeadm join 192.168.56.200:6443 --token vlybu0.e41soxtqv4muluua --discovery-token-ca-cert-hash sha256:44862b0c29b31870130ee6c994c9105bfae7ad553dff844e7929a6b291bf6ecf
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file /var/lib/kubelet/config.yaml
[kubelet-start] Writing kubelet environment file with flags to file /var/lib/kubelet/kubeadm-flags.env
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...
```

#### 4) 결과 확인하기

- w1 의 pod 가 생성됨을 확인할 수 있음

```
Every 2.0s: kubectl get pods,ds -o wide
k8s-m: Tue Ma

NAME                                READY    STATUS    RESTARTS   AGE    IP             NODE    NOMINATED NODE    READINESS GATES
pod/daemonset-1-8vlsr              0/1     ContainerCreating    0       7s     <none>         k8s-w1    <none>             <none>
pod/daemonset-1-zlvmv              1/1     Running           0       31m    172.30.46.61   k8s-w2    <none>             <none>

NAME                                DESIRED    CURRENT    READY    UP-TO-DATE    AVAILABLE    NODE SELECTOR    AGE    CONTAINERS    IMAGES    SELECTOR
daemonset.apps/daemonset-1          2          2          1        2              1            <none>           31m    daemonset-pod    nginx    name=daemonset-cloud
```

⇒ 아래와 같이 새로운 워커노드가 클러스터에 추가되면 daemonset 에 의해 지정된 파드가 자동으로 새 노드에 생성됨

⇒ 이처럼 daemonset 은 노드를 자동으로 배포 및 관리해줌

### 3-3. Tolerations 설정하기

```
# daemonset-2.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: daemonset-2
spec:
  selector:
    matchLabels:
      name: daemonset-cloud-2
```

```
template:
  metadata:
    labels:
      name: daemonset-cloud-2
  spec:
    tolerations:
      - effect: NoSchedule
        operator: Exists
    containers:
      - name: daemonset-pod-2
        image: nginx
```

1) 위와 같이 파일 만들고 적용하기

```
k apply -f daemonset-2.yaml
```

2) 모니터링으로 확인해보기

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
pod/daemonset-2-gx7zf	1/1	Running	0	10s	172.30.46.45	k8s-w2	<none>	<none>
pod/daemonset-2-j7bf8	1/1	Running	0	10s	172.30.228.112	k8s-w1	<none>	<none>
pod/daemonset-2-jcn86	0/1	ContainerCreating	0	10s	<none>	k8s-m	<none>	<none>

NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE	NODE SELECTOR	AGE	CONTAINERS	IMAGES	SELECTOR
daemonset.apps/daemonset-2	3	3	2	3	2	<none>	10s	daemonset-pod-2	nginx	name=daemonset-cloud-2

- 3-1.과 다르게 pod 가 3 개 생성됨! (마스터노드에도 생성됨, tolerations 설정으로 인해)

3) 삭제하기

```
k delete ds daemonset-2
```

## 4. Job

### 4-1. 파일 실행하기

```
# job.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    spec:
      containers:
        - name: pi
          image: perl
          command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"]
          restartPolicy: Never
      backoffLimit: 4
```

- 위의 코드는 원주율을 2000 자 까지 보여주라는 코드임

## 4-2. 모니터링하기

```
watch -d 'kubectl get pods,jobs -o wide'
```

## 4-3. 적용하기

```
kubectl apply -f job.yaml
```

- 모니터링으로 확인하기

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
pod/pi-67mm5	0/1	Completed	0	99s	172.30.46.46	k8s-w2	<none>	<none>

NAME	COMPLETIONS	DURATION	AGE	CONTAINERS	IMAGES	SELECTOR
job.batch/pi	1/1	99s	100s	pi	perl	batch.kubernetes.io/controller-uid=92d761f0-ef5a-4a03-88d7-94ca72ca88b6

## 4-4. 결과 확인하기

```
k logs -l job-name=pi
```

// 'job-name=pi'라벨을 가진 파드의 로그 조회하기

```
(calico:N/A) root@k8s-m:~# k logs -l job-name=pi
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803
5211055596446229489549303819644288109756659334461284756482337867831652712019091456485669
5364367892590360011330530548820466521384146951941511609433057270365759591953092186117381
4308602139494639522473719070217986094370277053921717629317675238467481846766940513200056
6892589235420199561121290219608640344181598136297747713099605187072113499999983729780499
8752886587533208381420617177669147303598253490428755468731159562863882353787593751957781
2796823030195203530185296899577362259941389124972177528347913151557485724245415069595082
5637076601047101819429555961989467678374494482553797747268471040475346462080466842590694
4199272604269922796782354781636009341721641219924586315030286182974555706749838505494588
0742654252786255181841757467289097777279380008164706001614524919217321721477235014144197
0992192221842725502542568876717904946016534668049886272327917860857843838279679766814541
6744278622039194945047123713786960956364371917287467764657573962413890865832645995813390
```

## 4-5. 이 상태에서 한 번 더 apply 하기

```
(calico:N/A) root@k8s-m:~# kubectl apply -f job.yaml
job.batch/pi unchanged
```

- but 생성되지 x

- 이전에 pod 가 생성되고 completed 되었지만 여전히 존재하고 있기 때문에 새로 생성 x

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
pod/pi-67mm5	0/1	Completed	0	99s	172.30.46.46	k8s-w2	<none>	<none>

NAME	COMPLETIONS	DURATION	AGE	CONTAINERS	IMAGES	SELECTOR
job.batch/pi	1/1	99s	100s	pi	perl	batch.kubernetes.io/controller-uid=92d761f0-ef5a-4a03-88d7-94ca72ca88b6

## 4-6. 삭제하기

```
k delete -f job.yaml
```

# 5. Cronjob

## 5-1. 파일 실행하기

```
# job.yaml
apiVersion: batch/v1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *" # Job 실행 주기 (1분에 한 번씩 작업 실행)
  successfulJobsHistoryLimit: 10 # 성공 Job 최대 기록
  jobTemplate: # 실행될 Job 설정 내용(spec)
    spec:
      template:
        spec:
          containers:
            - name: hello
              image: busybox
              imagePullPolicy: IfNotPresent
              command:
                - /bin/sh
                - -c
                - date; echo Hello from the BABO Member!
          restartPolicy: OnFailure
```

## 5-2. 모니터링하기

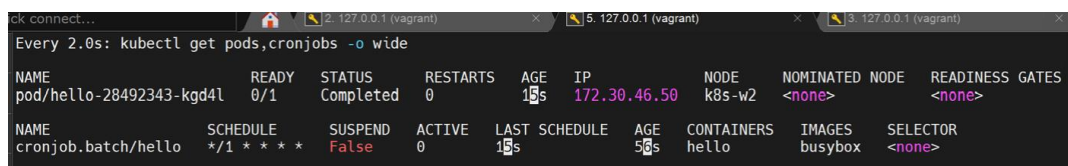
```
watch -d 'kubectl get pods,cronjobs -o wide'
```

## 5-3. 적용하기

```
k apply -f cronjob.yaml && k get jobs.batch -w
```

- 모니터링하기

◦ 원래는 아래와 같이 pod 1 개 었지만



Every 2.0s: kubectl get pods,cronjobs -o wide									
NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES	
pod/hello-28492343-kgd4l	0/1	Completed	0	15s	172.30.46.50	k8s-w2	<none>	<none>	
NAME	SCHEDULE	SUSPEND	ACTIVE	LAST SCHEDULE	AGE	CONTAINERS	IMAGES	SELECTOR	
cronjob.batch/hello	*/1 * * * *	False	0	15s	50s	hello	busybox	<none>	



- 적용하면 cronjob 이 60 초가 될 때마다 pod 가 1 개씩 생성됨

```

Every 2.0s: kubectl get pods,cronjobs -o wide
k8s-m: Mon Mar  4 17:24:19

NAME                READY   STATUS    RESTARTS   AGE   IP            NODE   NOMINATED NODE   READINESS GATES
pod/hello-28492343-kgd4l  0/1     Completed  0          79s   172.30.46.50  k8s-w2  <none>           <none>
pod/hello-28492344-m7xzk  0/1     Completed  0          19s   172.30.46.51  k8s-w2  <none>           <none>

NAME                SCHEDULE   SUSPEND   ACTIVE   LAST SCHEDULE   AGE   CONTAINERS   IMAGES   SELECTOR
cronjob.batch/hello  */1 * * * *  False    0        19s             2m    hello        busybox    <none>

```

- 나중에 확인하면 pod 여러 개가 계속 생성됨

```

(calico:N/A) root@k8s-m:~# k get pod
NAME                READY   STATUS    RESTARTS   AGE
hello-28492343-kgd4l  0/1     Completed  0          3m31s
hello-28492344-m7xzk  0/1     Completed  0          2m31s
hello-28492345-p8tlw  0/1     Completed  0          91s
hello-28492346-lrnmq  0/1     Completed  0          30s

```

## 5-4. log 확인하기

`watch -n 1 'kubectl logs -l job-name'`

```

Every 1.0s: kubectl logs -l job-name
Mon Mar  4 08:30:02 UTC 2024
Hello from the BABO Member!
Mon Mar  4 08:23:02 UTC 2024
Hello from the BABO Member!
Mon Mar  4 08:24:02 UTC 2024
Hello from the BABO Member!
Mon Mar  4 08:25:02 UTC 2024
Hello from the BABO Member!
Mon Mar  4 08:26:03 UTC 2024
Hello from the BABO Member!
Mon Mar  4 08:27:02 UTC 2024
Hello from the BABO Member!
Mon Mar  4 08:28:02 UTC 2024
Hello from the BABO Member!
Mon Mar  4 08:29:02 UTC 2024
Hello from the BABO Member!

```

- yaml 파일에서 date와 'Hello from the BAO Member!' 출력되게 설정했음

## 5-5. cronjob을 30초로 구현하기

- cronjob은 1분마다 실행되도록 설정할 수 있어 초 단위를 지원하지 x

⇒ sleep 기능을 활용하기!

### 1) yaml 파일 수정하기

```

# job-1.yaml
apiVersion: batch/v1
kind: CronJob

```



```

metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"      # Job 실행 주기
  successfulJobsHistoryLimit: 10 # 성공 Job 최대 기록
  jobTemplate:                 # 실행될 Job 설정 내용(spec)
    spec:
      template:
        spec:
          containers:
            - name: hello
              image: busybox
              imagePullPolicy: IfNotPresent
              command:
                - /bin/sh
                - -c
                - date; echo "Hello from the BABO Member!"; sleep 30; date; echo "Hello from the BABO Member 2!"
          restartPolicy: OnFailure

```

## 2) 적용 후 확인하기

```

Mon Mar  4 21:42:00 UTC 2024
Hello from the BABO Member!
Mon Mar  4 21:39:00 UTC 2024
Hello from the BABO Member!
Mon Mar  4 21:39:30 UTC 2024
Hello from the BABO Member 2!
Mon Mar  4 21:40:01 UTC 2024
Hello from the BABO Member!
Mon Mar  4 21:40:31 UTC 2024
Hello from the BABO Member 2!
Mon Mar  4 21:41:00 UTC 2024
Hello from the BABO Member!
Mon Mar  4 21:41:30 UTC 2024
Hello from the BABO Member 2!

```

- 30초마다 실행되긴 하는데 39:00 → 39:30 → 40:00 ... → 39:00로 돌아옴
- pod가 다시 생성되서 그런 것 같음.

## 출처

[https://www.google.com/search?q=coredns&oq=coredns&gs\\_lcrp=EgZjaHJvbWUyBggAEEUYOTIHCAEQABiABDIHCAIQLhiABDIHCAMQABiABDIGCAQRRg9MgYIBRBFGDwyBggGEEUYPDIGCAcQRRg80gEIMjM5OGowajeoAgCwAgA&sourceid=chrome&ie=UTF-8#ip=1](https://www.google.com/search?q=coredns&oq=coredns&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIHCAEQABiABDIHCAIQLhiABDIHCAMQABiABDIGCAQRRg9MgYIBRBFGDwyBggGEEUYPDIGCAcQRRg80gEIMjM5OGowajeoAgCwAgA&sourceid=chrome&ie=UTF-8#ip=1)

<https://nirsa.tistory.com/138> (데몬셋)

<https://etloveguitar.tistory.com/56> (데몬셋)

<https://kimjingo.tistory.com/146> (Taints & Toleration)

<https://malwareanalysis.tistory.com/151> (job)

재경님, 나경님 실습가이드