AIPod 구축을 위한 고성능 작업의 예

NetApp Solutions

NetApp
May 10, 2024

목차

AIPod 구축을 위한 고성능 작업의 예

단일 노드 AI 워크로드 실행

Kubernetes 클러스터에서 단일 노드 AI 및 ML 작업을 실행하려면 배포 점프 호스트에서 다음 작업을 수행하십시오. Trident를 사용하면 페타바이트에 이를 수 있는 데이터 볼륨을 빠르고 쉽게 만들어 Kubernetes 워크로드에 액세스할 수 있습니다. Kubernetes Pod에서 데이터 볼륨에 액세스할 수 있도록 하려면 POD 정의에 PVC를 지정하기만 하면 됩니다.

이 섹션에서는 Kubernetes 클러스터에서 실행하려고 하는 특정 AI 및 ML 워크로드를 이미 컨테이너화(Docker 컨테이너 형식)했다고 가정합니다.

1. 다음 명령 예는 ImageNet 데이터 세트를 사용하는 TensorFlow 벤치마크 워크로드에 대한 Kubernetes 작업 생성을 보여줍니다. ImageNet 데이터 세트에 대한 자세한 내용은 를 참조하십시오 "ImageNet 웹 사이트".

   이 예시 작업은 8개의 GPU를 요청하므로 8개 이상의 GPU를 갖춘 단일 GPU 작업자 노드에서 실행할 수 있습니다. 이 예시 작업은 8개 이상의 GPU를 갖춘 작업자 노드가 없거나 현재 다른 워크로드를 사용 중인 클러스터에 제출할 수 있습니다. 이 경우 해당 작업자 노드를 사용할 수 있을 때까지 작업은 보류 중 상태로 유지됩니다.

   또한 스토리지 대역폭을 최대화하기 위해 필요한 교육 데이터가 들어 있는 볼륨이 이 작업에서 생성되는 POD 내에 두 번 마운트됩니다. 포드에도 다른 볼륨이 마운트됩니다. 이 두 번째 볼륨은 결과 및 메트릭을 저장하는 데 사용됩니다. 이러한 용적은 PVC 이름을 사용하여 작업 정의에서 참조됩니다. Kubernetes 작업에 대한 자세한 내용은 를 참조하십시오 "Kubernetes 공식 문서".

   이 예시 작업이 생성하는 포드의 /dev/shm에 Memory의 midium 값을 가진 emptyDir 볼륨이 실장된다. Docker 컨테이너 런타임을 통해 자동으로 생성되는 '/dev/shm' 가상 볼륨의 기본 크기는 TensorFlow의 요구 사항에 비해 부족할 수 있습니다. 다음 예제와 같이 "emptyDir" 볼륨을 마운트하면 충분히 큰 "/dev/shm" 가상 볼륨이 제공됩니다. 'emptyDir' 볼륨에 대한 자세한 내용은 를 참조하십시오 "Kubernetes 공식 문서".

   이 예제 작업 정의에 지정된 단일 컨테이너에는 'ecurityContext > privileged' 값이 'true'로 지정됩니다. 이 값은 컨테이너가 호스트에 대한 루트 액세스 권한을 효과적으로 가지고 있음을 의미합니다. 이 경우 실행되는 특정 워크로드에 루트 액세스가 필요하므로 이 주석이 사용됩니다. 특히, 워크로드가 수행하는 명확한 캐시 작업에서는 루트 액세스가 필요합니다. 이 "특권" 주석이 필요한지 여부는 실행 중인 특정 워크로드의 요구 사항에 따라 달라집니다.

```
$ cat << EOF > ./netapp-tensorflow-single-imagenet.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: netapp-tensorflow-single-imagenet
spec:
  backoffLimit: 5
  template:
    spec:
      volumes:
      - name: dshm
        emptyDir:
```

```
          medium: Memory
      - name: testdata-iface1
        persistentVolumeClaim:
          claimName: pb-fg-all-iface1
      - name: testdata-iface2
        persistentVolumeClaim:
          claimName: pb-fg-all-iface2
      - name: results
        persistentVolumeClaim:
          claimName: tensorflow-results
      containers:
      - name: netapp-tensorflow-py2
        image: netapp/tensorflow-py2:19.03.0
        command: ["python", "/netapp/scripts/run.py", "--
dataset_dir=/mnt/mount_0/dataset/imagenet", "--dgx_version=dgx1", "--
num_devices=8"]
        resources:
          limits:
            nvidia.com/gpu: 8
        volumeMounts:
        - mountPath: /dev/shm
          name: dshm
        - mountPath: /mnt/mount_0
          name: testdata-iface1
        - mountPath: /mnt/mount_1
          name: testdata-iface2
        - mountPath: /tmp
          name: results
        securityContext:
          privileged: true
      restartPolicy: Never
EOF
$ kubectl create -f ./netapp-tensorflow-single-imagenet.yaml
job.batch/netapp-tensorflow-single-imagenet created
$ kubectl get jobs
NAME                                    COMPLETIONS   DURATION   AGE
netapp-tensorflow-single-imagenet       0/1           24s        24s
```

2. 1단계에서 만든 작업이 올바르게 실행 중인지 확인합니다. 다음 명령 예에서는 작업 정의에 지정된 대로 작업에 대해 단일 POD가 생성되었으며 이 POD가 현재 GPU 작업자 노드 중 하나에서 실행되고 있음을 확인합니다.

```
$ kubectl get pods -o wide
NAME                                                READY   STATUS
RESTARTS   AGE
IP              NODE            NOMINATED NODE
netapp-tensorflow-single-imagenet-m7x92          1/1     Running     0
3m    10.233.68.61    10.61.218.154   <none>
```

3. 1단계에서 생성한 작업이 성공적으로 완료되었는지 확인합니다. 다음 명령 예에서는 작업이 성공적으로 완료되었음을 확인합니다.

```
$ kubectl get jobs
NAME                                                COMPLETIONS   DURATION
AGE
netapp-tensorflow-single-imagenet                   1/1           5m42s
10m
$ kubectl get pods
NAME                                                      READY   STATUS
RESTARTS   AGE
netapp-tensorflow-single-imagenet-m7x92                   0/1     Completed
0          11m
$ kubectl logs netapp-tensorflow-single-imagenet-m7x92
[netapp-tensorflow-single-imagenet-m7x92:00008] PMIX ERROR: NO-
PERMISSIONS in file gds_dstore.c at line 702
[netapp-tensorflow-single-imagenet-m7x92:00008] PMIX ERROR: NO-
PERMISSIONS in file gds_dstore.c at line 711
Total images/sec = 6530.59125
================ Clean Cache !!! ==================
mpirun -allow-run-as-root -np 1 -H localhost:1 bash -c 'sync; echo 1 >
/proc/sys/vm/drop_caches'
==========================================
mpirun -allow-run-as-root -np 8 -H localhost:8 -bind-to none -map-by
slot -x NCCL_DEBUG=INFO -x LD_LIBRARY_PATH -x PATH python
/netapp/tensorflow/benchmarks_190205/scripts/tf_cnn_benchmarks/tf_cnn_be
nchmarks.py --model=resnet50 --batch_size=256 --device=gpu
--force_gpu_compatible=True --num_intra_threads=1 --num_inter_threads=48
--variable_update=horovod --batch_group_size=20 --num_batches=500
--nodistortions --num_gpus=1 --data_format=NCHW --use_fp16=True
--use_tf_layers=False --data_name=imagenet --use_datasets=True
--data_dir=/mnt/mount_0/dataset/imagenet
--datasets_parallel_interleave_cycle_length=10
--datasets_sloppy_parallel_interleave=False --num_mounts=2
--mount_prefix=/mnt/mount_%d --datasets_prefetch_buffer_size=2000
--datasets_use_prefetch=True --datasets_num_private_threads=4
--horovod_device=gpu >
/tmp/20190814_105450_tensorflow_horovod_rdma_resnet50_gpu_8_256_b500_ima
genet_nodistort_fp16_r10_m2_nockpt.txt 2>&1
```

4. * 선택 사항: * 작업 아티팩트 정리. 다음 예제 명령은 1단계에서 만든 작업 오브젝트의 삭제를 보여 줍니다.

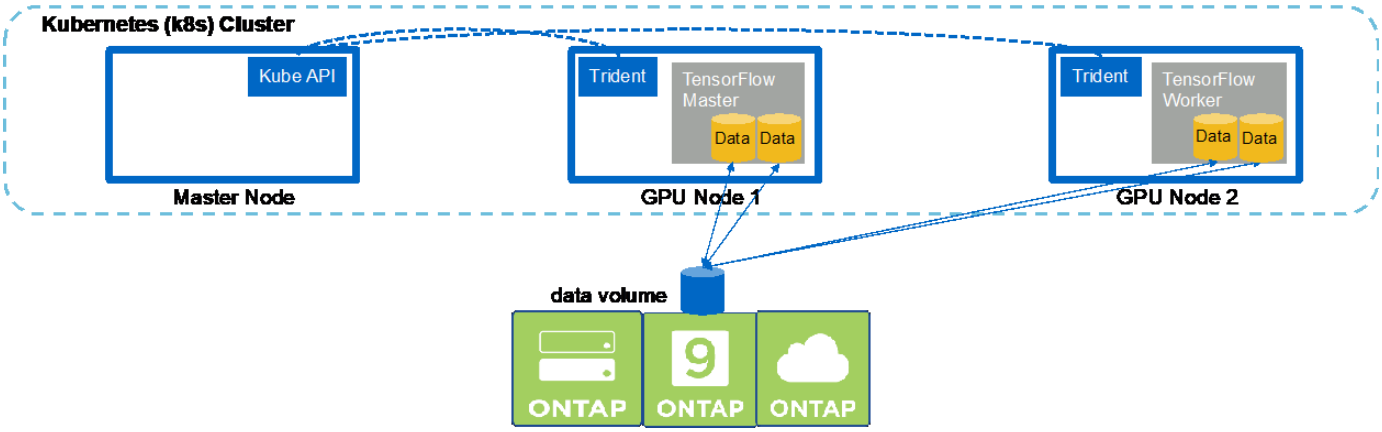
   작업 개체를 삭제하면 Kubernetes에서 연결된 포드를 자동으로 삭제합니다.

```
$ kubectl get jobs
NAME                                          COMPLETIONS   DURATION
AGE
netapp-tensorflow-single-imagenet             1/1           5m42s
10m
$ kubectl get pods
NAME                                                READY   STATUS
RESTARTS   AGE
netapp-tensorflow-single-imagenet-m7x92             0/1     Completed
0          11m
$ kubectl delete job netapp-tensorflow-single-imagenet
job.batch "netapp-tensorflow-single-imagenet" deleted
$ kubectl get jobs
No resources found.
$ kubectl get pods
No resources found.
```

동기식 분산 AI 워크로드 실행

Kubernetes 클러스터에서 동기식 다중 노드 AI 및 ML 작업을 실행하려면 배포 점프 호스트에서 다음 작업을 수행하십시오. 이 프로세스를 통해 NetApp 볼륨에 저장된 데이터를 활용하고 단일 작업자 노드가 제공할 수 있는 것보다 더 많은 GPU를 사용할 수 있습니다. 동기식 분산 AI 작업을 설명하는 방법은 다음 그림을 참조하십시오.

동기 분산 작업은 비동기 분산 작업에 비해 성능 및 교육 정확도를 높일 수 있습니다. 동기 작업과 비동기 작업의 장단점을 논하는 것은 이 문서의 범위를 벗어납니다.



1. 다음 명령 예는 섹션의 예에서 단일 노드에서 실행된 동일한 TensorFlow 벤치마크 작업의 동기식 분산 실행에 참여하는 작업자 1명의 생성을 보여 줍니다 "단일 노드 AI 워크로드 실행". 이 특정 예제에서는 작업이 두 작업자 노드에 걸쳐 실행되므로 한 명의 작업자만 배포됩니다.

   이 작업자 배포는 8개의 GPU를 요청하므로 8개 이상의 GPU를 갖춘 단일 GPU 작업자 노드에서 실행할 수 있습니다. GPU 작업자 노드에서 8개 이상의 GPU를 사용하여 성능을 극대화한 경우, 이 숫자를 작업자 노드가 갖춘

GPU 수와 같게 늘리고 싶을 수 있습니다. Kubernetes 구축에 대한 자세한 내용은 를 참조하십시오 "Kubernetes 공식 문서".

이 예에서는 특정 컨테이너형 작업자가 자체적으로 완료되지 않기 때문에 Kubernetes 구축이 생성됩니다. 따라서 Kubernetes 작업 구성을 사용하여 구축하는 것은 타당하지 않습니다. 작업자가 혼자서 완료되도록 설계되거나 작성된 경우 작업 구성을 사용하여 작업자를 배포하는 것이 합리일 수 있습니다.

이 예제 배포 사양에 지정된 POD에 "true"의 "hostNetwork" 값이 제공됩니다. 이 값은 이 Pod가 일반적으로 Kubernetes에서 각 Pod에 생성하는 가상 네트워킹 스택 대신 호스트 작업자 노드의 네트워킹 스택을 사용한다는 것을 의미합니다. 이 경우 특정 워크로드는 개방형 MPI, NCCL 및 Horovod를 통해 동기식 분산 방식으로 워크로드를 실행하기 때문에 이 주석이 사용됩니다. 따라서 호스트 네트워킹 스택에 액세스해야 합니다. 공개 MPI, NCCL 및 Horovod에 대한 논의는 이 문서의 범위를 벗어납니다. 이 "hostNetwork: true" 주석이 필요한지 여부는 실행 중인 특정 워크로드의 요구 사항에 따라 달라집니다. hostNetwork 필드에 대한 자세한 내용은 를 참조하십시오 "Kubernetes 공식 문서".

```
$ cat << EOF > ./netapp-tensorflow-multi-imagenet-worker.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: netapp-tensorflow-multi-imagenet-worker
spec:
  replicas: 1
  selector:
    matchLabels:
      app: netapp-tensorflow-multi-imagenet-worker
  template:
    metadata:
      labels:
        app: netapp-tensorflow-multi-imagenet-worker
    spec:
      hostNetwork: true
      volumes:
      - name: dshm
        emptyDir:
          medium: Memory
      - name: testdata-iface1
        persistentVolumeClaim:
          claimName: pb-fg-all-iface1
      - name: testdata-iface2
        persistentVolumeClaim:
          claimName: pb-fg-all-iface2
      - name: results
        persistentVolumeClaim:
          claimName: tensorflow-results
      containers:
      - name: netapp-tensorflow-py2
        image: netapp/tensorflow-py2:19.03.0
        command: ["bash", "/netapp/scripts/start-slave-multi.sh",
```

```
"22122"]
        resources:
          limits:
            nvidia.com/gpu: 8
        volumeMounts:
        - mountPath: /dev/shm
          name: dshm
        - mountPath: /mnt/mount_0
          name: testdata-iface1
        - mountPath: /mnt/mount_1
          name: testdata-iface2
        - mountPath: /tmp
          name: results
        securityContext:
          privileged: true
EOF
$ kubectl create -f ./netapp-tensorflow-multi-imagenet-worker.yaml
deployment.apps/netapp-tensorflow-multi-imagenet-worker created
$ kubectl get deployments
NAME                                        DESIRED   CURRENT   UP-TO-DATE
AVAILABLE   AGE
netapp-tensorflow-multi-imagenet-worker   1         1         1
1           4s
```

2. 1단계에서 만든 작업자 배포가 성공적으로 시작되었는지 확인합니다. 다음 예제 명령은 배포 정의에 나와 있는 것처럼 단일 작업자 POD가 배포용으로 생성되었으며 이 POD가 현재 GPU 작업자 노드 중 하나에서 실행되고 있음을 확인합니다.

```
$ kubectl get pods -o wide
NAME                                                              READY
STATUS    RESTARTS   AGE
IP              NODE            NOMINATED NODE
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725   1/1
Running   0          60s   10.61.218.154   10.61.218.154   <none>
$ kubectl logs netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725
22122
```

3. 동기 다중 노드 작업의 실행을 종료, 참여 및 추적하는 마스터에 대한 Kubernetes 작업을 생성합니다. 다음 명령 예에서는 이 섹션의 예제에서 단일 노드에서 실행된 것과 동일한 TensorFlow 벤치마크 작업의 동기식 분산 실행을 시작, 참여 및 추적하는 하나의 마스터를 생성합니다 "단일 노드 AI 워크로드 실행".

   이 마스터 작업에서는 8개의 GPU를 요청하므로 8개 이상의 GPU를 갖춘 단일 GPU 작업자 노드에서 실행할 수 있습니다. GPU 작업자 노드에서 8개 이상의 GPU를 사용하여 성능을 극대화한 경우, 이 숫자를 작업자 노드가 갖춘 GPU 수와 같게 늘리고 싶을 수 있습니다.

   이 예에서 지정한 마스터 포드는 1단계에서 작업자 포드가 hostNetwork 값이 true인 것처럼 true의 hostNetwork

값이 지정됩니다. 이 값이 필요한 이유에 대한 자세한 내용은 1단계를 참조하십시오.

```
$ cat << EOF > ./netapp-tensorflow-multi-imagenet-master.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: netapp-tensorflow-multi-imagenet-master
spec:
  backoffLimit: 5
  template:
    spec:
      hostNetwork: true
      volumes:
      - name: dshm
        emptyDir:
          medium: Memory
      - name: testdata-iface1
        persistentVolumeClaim:
          claimName: pb-fg-all-iface1
      - name: testdata-iface2
        persistentVolumeClaim:
          claimName: pb-fg-all-iface2
      - name: results
        persistentVolumeClaim:
          claimName: tensorflow-results
      containers:
      - name: netapp-tensorflow-py2
        image: netapp/tensorflow-py2:19.03.0
        command: ["python", "/netapp/scripts/run.py", "--
dataset_dir=/mnt/mount_0/dataset/imagenet", "--port=22122", "--
num_devices=16", "--dgx_version=dgx1", "--
nodes=10.61.218.152,10.61.218.154"]
        resources:
          limits:
            nvidia.com/gpu: 8
        volumeMounts:
        - mountPath: /dev/shm
          name: dshm
        - mountPath: /mnt/mount_0
          name: testdata-iface1
        - mountPath: /mnt/mount_1
          name: testdata-iface2
        - mountPath: /tmp
          name: results
        securityContext:
          privileged: true
```

```
      restartPolicy: Never
EOF
$ kubectl create -f ./netapp-tensorflow-multi-imagenet-master.yaml
job.batch/netapp-tensorflow-multi-imagenet-master created
$ kubectl get jobs
NAME                                        COMPLETIONS   DURATION   AGE
netapp-tensorflow-multi-imagenet-master   0/1           25s        25s
```

4. 3단계에서 만든 마스터 작업이 올바르게 실행되고 있는지 확인합니다. 다음 예제 명령은 작업 정의에 나와 있는 것처럼 작업에 대해 단일 마스터 포드가 생성되었으며 이 포드가 현재 GPU 작업자 노드 중 하나에서 실행되고 있음을 확인합니다. 또한 1단계에서 처음 보았던 작업자 포드가 여전히 실행 중이고 마스터 포드와 작업자 포드가 다른 노드에서 실행되고 있음을 확인해야 합니다.

```
$ kubectl get pods -o wide
NAME                                                              READY
STATUS    RESTARTS   AGE
IP              NODE            NOMINATED NODE
netapp-tensorflow-multi-imagenet-master-ppwwj                     1/1
Running   0          45s   10.61.218.152   10.61.218.152   <none>
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725   1/1
Running   0          26m   10.61.218.154   10.61.218.154   <none>
```

5. 3단계에서 만든 마스터 작업이 성공적으로 완료되었는지 확인합니다. 다음 명령 예에서는 작업이 성공적으로 완료되었음을 확인합니다.

```
$ kubectl get jobs
NAME                                        COMPLETIONS   DURATION   AGE
netapp-tensorflow-multi-imagenet-master   1/1           5m50s      9m18s
$ kubectl get pods
NAME                                                              READY
STATUS      RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj                     0/1
Completed   0          9m38s
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725   1/1
Running     0          35m
$ kubectl logs netapp-tensorflow-multi-imagenet-master-ppwwj
[10.61.218.152:00008] WARNING: local probe returned unhandled
shell:unknown assuming bash
rm: cannot remove '/lib': Is a directory
[10.61.218.154:00033] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 702
[10.61.218.154:00033] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 711
[10.61.218.152:00008] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 702
```

```
[10.61.218.152:00008] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 711
Total images/sec = 12881.33875
================ Clean Cache !!! ==================
mpirun -allow-run-as-root -np 2 -H 10.61.218.152:1,10.61.218.154:1 -mca
pml ob1 -mca btl ^openib -mca btl_tcp_if_include enp1s0f0 -mca
plm_rsh_agent ssh -mca plm_rsh_args "-p 22122" bash -c 'sync; echo 1 >
/proc/sys/vm/drop_caches'
=========================================
mpirun -allow-run-as-root -np 16 -H 10.61.218.152:8,10.61.218.154:8
-bind-to none -map-by slot -x NCCL_DEBUG=INFO -x LD_LIBRARY_PATH -x PATH
-mca pml ob1 -mca btl ^openib -mca btl_tcp_if_include enp1s0f0 -x
NCCL_IB_HCA=mlx5 -x NCCL_NET_GDR_READ=1 -x NCCL_IB_SL=3 -x
NCCL_IB_GID_INDEX=3 -x
NCCL_SOCKET_IFNAME=enp5s0.3091,enp12s0.3092,enp132s0.3093,enp139s0.3094
-x NCCL_IB_CUDA_SUPPORT=1 -mca orte_base_help_aggregate 0 -mca
plm_rsh_agent ssh -mca plm_rsh_args "-p 22122" python
/netapp/tensorflow/benchmarks_190205/scripts/tf_cnn_benchmarks/tf_cnn_be
nchmarks.py --model=resnet50 --batch_size=256 --device=gpu
--force_gpu_compatible=True --num_intra_threads=1 --num_inter_threads=48
--variable_update=horovod --batch_group_size=20 --num_batches=500
--nodistortions --num_gpus=1 --data_format=NCHW --use_fp16=True
--use_tf_layers=False --data_name=imagenet --use_datasets=True
--data_dir=/mnt/mount_0/dataset/imagenet
--datasets_parallel_interleave_cycle_length=10
--datasets_sloppy_parallel_interleave=False --num_mounts=2
--mount_prefix=/mnt/mount_%d --datasets_prefetch_buffer_size=2000 --
datasets_use_prefetch=True --datasets_num_private_threads=4
--horovod_device=gpu >
/tmp/20190814_161609_tensorflow_horovod_rdma_resnet50_gpu_16_256_b500_im
agenet_nodistort_fp16_r10_m2_nockpt.txt 2>&1
```

6. 작업자 배포가 더 이상 필요하지 않으면 삭제합니다. 다음 예제 명령은 1단계에서 만든 작업자 배포 개체를 삭제하는 방법을 보여 줍니다.

   작업자 배포 개체를 삭제하면 Kubernetes에서 연결된 작업자 포드를 자동으로 삭제합니다.

```
$ kubectl get deployments
NAME                                          DESIRED   CURRENT   UP-TO-DATE
AVAILABLE   AGE
netapp-tensorflow-multi-imagenet-worker   1         1         1
1           43m
$ kubectl get pods
NAME                                                              READY
STATUS      RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj                     0/1
Completed   0          17m
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725   1/1
Running     0          43m
$ kubectl delete deployment netapp-tensorflow-multi-imagenet-worker
deployment.extensions "netapp-tensorflow-multi-imagenet-worker" deleted
$ kubectl get deployments
No resources found.
$ kubectl get pods
NAME                                               READY   STATUS
RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj   0/1     Completed   0
18m
```

7. * 선택 사항: * 마스터 작업 아티팩트를 정리하십시오. 다음 예제 명령은 3단계에서 만든 마스터 작업 오브젝트의 삭제를 보여 줍니다.

   마스터 작업 개체를 삭제하면 연결된 마스터 포드가 자동으로 삭제됩니다.

```
$ kubectl get jobs
NAME                                       COMPLETIONS   DURATION   AGE
netapp-tensorflow-multi-imagenet-master   1/1           5m50s      19m
$ kubectl get pods
NAME                                             READY   STATUS
RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj   0/1     Completed   0
19m
$ kubectl delete job netapp-tensorflow-multi-imagenet-master
job.batch "netapp-tensorflow-multi-imagenet-master" deleted
$ kubectl get jobs
No resources found.
$ kubectl get pods
No resources found.
```