For your lab demonstrating two of the most widely used **Kubernetes Job patterns**, we'll focus on these patterns:

1. **Single-Completion Job** – where one task runs to completion.
2. **Parallel Job with Multiple Completions** – where multiple pods run in parallel, and all must complete successfully for the job to finish.

Here's how you can structure a realistic lab setup for these two patterns. Each Job will perform meaningful tasks, such as simulating data processing and distributed work.

Lab Setup Overview

- **Scenario 1: Single-Completion Job**
  This will simulate a one-time data migration task, where a single pod processes a data file and moves it to a different location.
- **Scenario 2: Parallel Job with Multiple Completions**
  This will simulate distributed data processing, where multiple pods each process a chunk of a large dataset in parallel.

Preparation:

Create a directory for the lab:

```
mkdir jobs
```

1. Scenario 1: Single-Completion Job

Use Case: Data Migration Job

In this scenario, we simulate a single-completion task where a container pulls a data file from one location (simulating an S3 bucket or database), processes it (e.g., filters, cleans), and stores the processed data in another location.

Job Manifest for Scenario 1:

```
# single-job.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: data-migration-job
spec:
  completions: 1
  parallelism: 1
  template:
    spec:
      containers:
      - name: migration
        image: busybox
        command: ["sh", "-c", "echo 'Migrating data...' && sleep 10 && echo 'Data migration
complete.'"]
      restartPolicy: Never
```

Explanation:

- **completions**: 1 ensures the job runs to completion once.
- **parallelism**: 1 ensures that only one pod runs at a time.
- The task involves simulating a data migration process using a simple `echo` and `sleep` command.

Steps to Run:

1. Apply the Job manifest:

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

2. Check the Job status:

   ```
   kubectl get jobs
   ```

3. View the logs of the Job pod:

   ```
   kubectl get pods
   kubectl logs pod_name
   kubectl logs pod_name -f
   ```

You will see output like:

```
Migrating data...
Data migration complete.
```

This simulates the data migration being performed by a single job.

Run

```
kubectl get jobs
```

You should see that the pod is marked as `completed`.

---

2. Scenario 2: Parallel Job with Multiple Completions

Use Case: Distributed Data Processing Job

In this scenario, multiple pods will each process a different part of a dataset in parallel. This could be part of a MapReduce-like operation where each pod handles a specific chunk of data and the task is complete only when all the chunks have been processed.

Job Manifest for Scenario 2:

```
apiVersion: batch/v1
kind: Job
metadata:
  name: data-processing-job
spec:
  completions: 5
  parallelism: 3
  template:
    spec:
      containers:
      - name: processor
        image: busybox
        command: ["sh", "-c", "echo 'Processing chunk $(MY_CHUNK)...' && sleep 20 && echo 'Chunk
$(MY_CHUNK) processed.'"]
        env:
        - name: MY_CHUNK
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
      restartPolicy: Never
```

Explanation:

- **completions**: 5 ensures that the Job is complete only after 5 pods (or instances) have finished.
- **parallelism**: 3 ensures that 3 pods will run simultaneously, processing different chunks.
- **environment variable**: Each pod will get its unique name using `metadata.name`, and the pod will process a simulated "chunk" of data.
- The task involves simulating a data processing job using a simple `echo` and `sleep` command.

Steps to Run:

1. Apply the Job manifest:

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

2. View the logs of each Job pod (repeat this for all running/complete pods):

   ```
   kubectl get pods # there should be three
   kubectl pod_name # (for one of the three pods)
   kubectl get pods # (they should be still running till the fourth and fifth ones are
   complete)
   ```

You will see logs from each pod showing its progress:

```
Processing chunk data-processing-job-...
Chunk data-processing-job- processed.
```

In this scenario, 5 chunks are processed in total, and 3 of them are processed in parallel.

---

4. Cleanup

After the lab, clean up the resources by deleting the Jobs:

```
kubectl delete job data-migration-job
kubectl delete job data-processing-job
```

This will ensure that no resources are left running in the cluster.