**1. What does a SavedModel contain? How do you inspect its content?**

* **SavedModel**:
  + Contains the model's architecture, weights, and metadata.
  + Includes a saved\_model.pb file (model architecture) and a variables directory (weights).
* **Inspect content**:
  + Use the saved\_model\_cli tool:

saved\_model\_cli show --dir /path/to/saved\_model --all

**2. When should you use TF Serving? What are its main features? What are some tools you can use to deploy it?**

* **When to use**:
  + When you need to serve TensorFlow models in production with high performance and scalability.
* **Main features**:
  + Supports versioning and automatic model updates.
  + Provides REST and gRPC APIs for inference.
  + Optimized for low-latency, high-throughput serving.
* **Deployment tools**:
  + Docker, Kubernetes, and TensorFlow Serving's native binaries.

**3. How do you deploy a model across multiple TF Serving instances?**

* Use a **load balancer** (e.g., NGINX, HAProxy) to distribute requests across multiple TF Serving instances.
* Ensure all instances have access to the same model files (e.g., via a shared file system or cloud storage).

**4. When should you use the gRPC API rather than the REST API to query a model served by TF Serving?**

* Use the **gRPC API**:
  + When you need lower latency and higher throughput.
  + For binary data or large payloads.
  + When you require bidirectional streaming or advanced features like authentication.

**5. What are the different ways TFLite reduces a model's size to make it run on a mobile or embedded device?**

* **Quantization**: Reduces precision of weights and activations (e.g., from 32-bit floats to 8-bit integers).
* **Pruning**: Removes less important weights or neurons.
* **Weight clustering**: Groups similar weights together to reduce redundancy.
* **Model optimization**: Simplifies the model architecture (e.g., reducing layers or parameters).

**6. What is quantization-aware training, and why would you need it?**

* **Quantization-aware training**:
  + A technique where the model is trained with simulated quantization to minimize accuracy loss when the model is quantized.
* **Why use it**:
  + Ensures the model performs well after quantization, which is critical for deployment on resource-constrained devices.

**7. What are model parallelism and data parallelism? Why is the latter generally recommended?**

* **Model parallelism**:
  + Splits the model across multiple devices, with each device handling a part of the model.
* **Data parallelism**:
  + Replicates the model on multiple devices, with each device processing a different subset of the data.
* **Why data parallelism is preferred**:
  + Easier to implement and scale.
  + Works well with large datasets and distributed training.

**8. When training a model across multiple servers, what distribution strategies can you use? How do you choose which one to use?**

* **Distribution strategies**:
  + **MirroredStrategy**: Synchronous training across multiple GPUs on a single machine.
  + **MultiWorkerMirroredStrategy**: Synchronous training across multiple machines.
  + **ParameterServerStrategy**: Asynchronous training with parameter servers.
  + **TPUStrategy**: Training on TPUs.
* **Choosing a strategy**:
  + Use **MirroredStrategy** for single-machine multi-GPU training.
  + Use **MultiWorkerMirroredStrategy** for distributed training across multiple machines.
  + Use **ParameterServerStrategy** for large-scale asynchronous training.
  + Use **TPUStrategy** for training on TPUs.