Note: acks = all must be used in conjunction with min.insync.replicas
lets say brokers = 3, min.insync = 2, acks = all.
=> you can tolerate only 1 broker going down. Minimum 2 replicas/brokers should respond that they have data. Otherwise error is produced in producer.send().
latency + safety acks = all (leader + replicas)

little latency, but if acks = 1 (leader)
replica go down before leader send them data. then data loss
potential data loss. acks = 0 (none)
(logs, metrics - use cases)

Producer retries:
Retry to send data n number of times.
- default n = 0.
- You can increase to a high number. eg Integer.max-value.

- In case of retries, messages may be sent out of order.
- If you rely on key-based ordering that can be huge issue.
- for this you can set the setting while controls how many produce request can be made in parallel

max.in.flight.requests.per.connection
default: 5
set it to 1 if you need to ensure ordering (may impact throughput)

If kafka > 1.0.0 there is better solution

Idempotent Producer
Producer can introduce duplicate messages in kafka due to network errors. (ack never reaches producer)

The solution is idempotent producer.
This will not create duplicate commits to kafka.

- Guarantees safe and stable pipeline
Idempotent Producer:
retries: Integer.max-value
max.in.flight.requests = 1 (kafka >= 0.11 & < 1.1) or
" " = 5 (kafka >= 1.1 - higher performance)
- acks = all.

Just set

producerProps.put("enable.idempotence", true);

Safe producer Summary & Demo

Kafka < 0.11

- acks=all (producer level)
  - Ensures data is properly replicated before an ack is received
- min.insync.replicas=2 (broker/topic level)
  - Ensures two brokers in ISR at least have the data after an ack
- retries=MAX_INT (producer level)
  - Ensures transient errors are retried indefinitely
- max.in.flight.requests.per.connection=1 (producer level)
  - Ensures only one request is tried at any time, preventing message re-ordering in case of retries

Kafka >= 0.11

- enable.idempotence=true (producer level) + min.insync.replicas=2 (broker/topic level)
  - Implies acks=all, retries=MAX_INT, max.in.flight.requests.per.connection=5 (default)
  - while keeping ordering guarantees and improving performance!

- Running a "safe producer" might impact throughput and latency, always test for your use case

So running a safe producer, though, I may say,

MacBook Air

Message compression

- Compression is enabled at Producer level and doesn't require any configuration change in the Brokers or Consumers

- compression.type can be 'none' (default), 'gzip', 'lz4', snappy

- It is more effective the bigger the batch of the messages being sent to kafka.

Adv:

- much smaller producer request size (compression ratio upto 4x!)
- Faster transfer of data => less latency
- Better throughput
- Better disk utilisation (stored msg on disk are smaller)

Disadv:

- Producer and consumers must commit some CPU cycles to compress/decompress.

Overall:

consider testing snappy or lz4 for optimal speed/compression ratio.

- Always use compression in production and especially if you have high throughput.

- Consider tweaking linger.ms and batch.size to have bigger batches and therefore more compression and more throughput.

Producer Batching

By default kafka tries to send records as soon as possible

- It will have upto 5 requests in flight, meaning upto 5 messages individually sent at a time.
- After this if more msgs have to be sent while others are in flight, kafka is smart and will start batching them while they wait to send them all at once.

- This smart batching allows kafka to increase throughput while maintaining low latency.

- Batches have high compression ratio so better

efficiency.

linger.ms : no. of ms producer will wait before sending a batch out
default : 0.

- By introducing some lag (linger.ms = 5) we can increase the chances of messages being sent together in a batch.

- So at expense of a small delay, we have high throughput, compression and efficiency of our producer.

- If batch is full before end of linger.ms period it will be sent to kafka right away.

batch.size - max no. of bytes that will be included in a batch
Default : 16 KB.

- Increasing to 32 KB or 64 KB can help increasing the compression, throughput and efficiency of requests.

- Any msg bigger than batch size will not be batched.

- Dont set it too high in size, as it is per partition and you will waste memory that way.

(You can monitor avg batch size metric using kafka producer metrics).