What type of queues can you create? State their differences.

There are two types of Amazon SQS queues: first-in, first-out (FIFO) and standard queues. In FIFO queues, message strings remain in the same order in which the original messages were sent and received. They can support up to 300 send, receive or delete messages per second. They are designed for messaging between applications where the order of operations and events is of paramount importance.
Standard queues attempt to keep message strings in the same order in which the messages were originally sent, but the original order or sequence of messages may change due to processing requirements. For example, standard queues can be used to batch messages for future processing or allocate tasks to multiple worker nodes.
The frequency of message delivery differs between standard and FIFO queues, as FIFO messages are delivered exactly once, while in standard queues, messages are delivered at least once.
There is no limit on the number of messages that any individual Amazon SQS message queue can contain. However, the system has different quotas for *inflight* messages for its two different queue types. A message is considered *inflight* once a consuming component has received it from the queue, but it has not yet been erased from the queue. Amazon SQS allows 120,000 inflight messages in standard queues and 20,000 in FIFO queues. There is no limit to the number of message queues a user can create, but the name of the message queue can be no longer than 80 characters.

In which situations is a Web-Queue-Worker architecture relevant?

This architecture style is relevant for a purely PaaS solution. In this style, the application has a web front end that handles HTTP requests and a back-end worker that performs CPU-intensive tasks or long-running operations. The front end is decoupled from the worker so they can be scaled independently and communicates to the worker through an asynchronous message queue.
In this case, Web-queue-worker is suitable for :

- applications with a relatively simple domain with some resource-intensive tasks (because with complex domains, it can be hard to manage dependencies)
- applications with some long-running workflows or batch operations
- when you want to use managed services, rather than infrastructure as a Service.