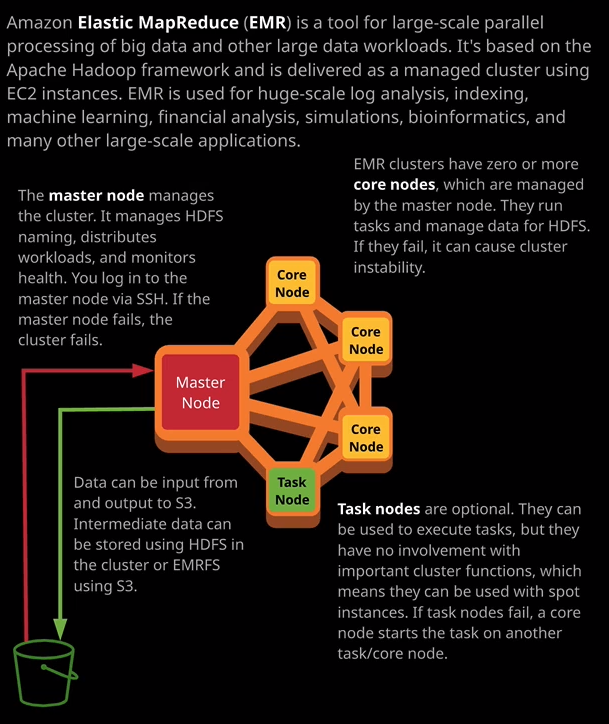
So those three products are going to be **EMR or Elastic MapReduce, Kinesis, and Kinesis Firehose** and then **Red Shift**. Now these products are things that I don't expect to feature in the exam in any great detail, but I want you to be aware of them to help you eliminate any incorrect answers for the exam, because that's often just as important as being able to identify the correct answer. So we're going to start off by talking about Elastic MapReduce, and this is going to be an entirely theory based lesson. So my apologies in advance, but I'll try to make it as brief as possible.



Now **EMR or Elastic MapReduce is a product which allows you to perform analysis of large scale, semi structured and unstructured data. It's often used when you're analyzing so called big data data sets. It's based on the Apache Hadoop ecosystem of products, but it's delivered as a managed service**. Now, I don't really expect you to have any questions which ask in detail about EMR. So it was considering no even having a lesson on it in the course but as always, I think it's better to have some knowledge on a product, even if it's just high level architecture, you never know when that little piece of background information we'll let you answer a seemingly unrelated question correctly.

**Now EMR is a cluster based product. You create an EMR cluster and within that are a few different types of nodes which run on EC2 instances that are managed by the product. So you're not going to get to manage individual EC2 instances, but the cluster that you create inside EMR will itself create and manage EC2 instances**. Now, before I do step through the node architecture of EMR, let's look at what EMR does exactly. **It processes huge sets of data. The cluster, at a really simplistic level, splits the work up and allocates portions of that to nodes, so you might find that your imput to an EMR cluster is stored on S3. The master node pulls that data in splits that up across the other types of cluster nodes, performs the processing and then it might put that data back onto S3. Because it's a cluster it needs a shared file system and traditionally, in Hadoop this was HDFS. It was a file system that exists for the lifetime of the cluster. So data is loaded into the cluster, stored on HDFS, process by it, and then moved off when the job is finished.   
Now, EMRFS is an enhancement of this, and this is specific EMR, and it's a cluster file system that's based on S3 so it's persistent.**   
**Data exists on the EMRFS even if the cluster is unreliable for some reason. So you get a lot of advantages to using EMRFS over HDFS for any work loads on EMR**. Now, the reason why this is important is that **one of the benefits of EMR is that it can be used for ad hoc processing tasks. Traditionally, Hadoop clusters used physical infrastructure and they took time and effort to set up. You generally create them and they operate for years. EMR allows you to spin up a cluster for a specific thing, do that thing, and then tear it down afterwards.** Now an example of this that you might come across is that if you use data pipeline to back up a DynamoDB table to S3 or the inverse of that so you restore data from S3 to a DynamoDB table, then data pipeline orchestrates the creation of an EMR cluster and that EMR cluster exists to perform that one single task before it's terminated. Now, EMR clusters can be long lived or short term or ad hoc. It depends on exactly what need you have as an architect.

**Now, within an EMR cluster, you've got three different types of nodes. Every cluster has a master node and the master told controls almost everything that happens in a cluster. It's the component which takes a job and splits that job across other nodes for the actual processing. It's the node, which managers HDFS naming. So if the master node fails, the cluster fails, and the file system fails. If you need to SSH into a cluster, it's the master node that you'll connect to. Master nodes therefore almost never use spot instances. If the master node is a spot instance then generally every other node type in the cluster should also be a spot instance because if the master nodes fails or it gets terminated than everything else in the cluster terminates along with it. Now, generally, the master node will only be a spot instance for short lived or ad hoc clusters. Normally, it's going to be using on demand or reserved.** Now moving on from the master node.   
Next, we've got **core nodes and core nodes serve two main functions. If you're using HDFS then they're managing the data for the HDFS shared file system. So if you do use that and a core node fails, then it can impact the storage as well as managing the storage, they also run jobs. If a core node fails, it can impact the cluster in two ways. One it can cause storage instability but also, if a core node fails, any task running on that node could fail and the overall job could fail.** So again, ideally, **core nodes are going to be running on stable compute so probably using on demand. They're only going to be using spot instances if the whole cluster is using spot instances because it's an ad hoc cluster**.   
Now, task nodes are the **third type of nodes, and they're entirely optional. They only run tasks. They have no involvement in any cluster operations. If they fail the cluster is designed to work through it. Task nodes often use spot instances, regardless of what the rest of the cluster is using because the cluster as a whole is designed to assume that if anything fails on a task node it could just pass that job off to another node and continue processing. So whenever you've got any sort of variable workloads where you want to utilize spot instances to get the best economy for any job that you're running on a cluster, then you should absolutely be using task nodes and be using spot instances for those task nodes and that goes whether it's an ad hoc EMR cluster or a long running EMR cluster. You can use spot instances and task nodes to get the best economy.**

Now S3 functions as a data store within EMR. It's either used as an input and output location which is used before and after a job starts and ends. So if you're using HDFS internally, then S3 is basically used to load the initial data set from and to store the final result to but if you're using EMRFS then the whole cluster file system is backed by S3, so it's used as a foundation for EMRFS. Now this is another product that I don't expect will feature heavily on the exam. What I want to do in this lesson is focus really tightly on the things that you might face in the exam. So if you do get any questions which involved EMR even to a tiny degree, it'll probably test your ability to know when to use EMR. So the crucial thing is when to use it. So **EMRs benefits is it's on demand billing so you can spin up an EMR cluster which is, after all, a managed implementation of Hadoop for short term or ad hoc task. So whenever you need to perform any semi structured or unstructured analysis of data then potentially, you can use Hadoop. The difference between EMR and Athena is that Athena is general use for querying data that's already on S3, whereas EMR is used for large scale data analytics. So if you've got any processing work loads where you need to manipulate data or do any calculations on data, perform Big Data Analytics, then I tend to prefer EMR versus Athena. If it's just serverless querying then Athena is probably going to be preferred to EMR.** Now remember that while **EMR is ideally suited for short term or ad hoc processing tasks, it can also be used for long running clusters. So weeks or months and if you do do that, then you can use reserved instances to achieve better economies of scale**.