**Instructions: FE to memorize answers to 30 questions a week starting:**

1. **During Training, as soon as the topic is covered**
2. **After Training, any remaining concepts that weren’t cover**

**What is Hadoop (HDFS, Yarn, Map Reduce)?**

* Framework to process large amounts of data.
* Note Hadoop and Big Data are not one in the same.
* Hadoop is only one solution or framework to process large amounts of data:
* Map Reduce (sits on top of)
* Yarn (sits on top of)
* HDFS
* There are three types of data: unstructured, semi-structured, and structured.

**What is Big Data?**

* A set of methodologies, tools, and frameworks to handle large amounts of information or data, as well (like Hadoop), but it's not restricted to one solutions.
* Organizations use Big Data to analyze extremely large data sets
* They do this to reveal patterns, find trends, make predictions, and correlate associations
* Data volumes can get up to 40 Zettabytes - That’s equivalent to adding every single grain of sand on the planet multiplied by 75

**Use Cases for Big Data**

A screenshot of a cell phone

Description generated with very high confidence

**Example of Structured, Unstructured, and Semi-Structured data**

* Structured: Excel (columns and well defined - you know what goes inside each column), or Databases
* Semi-Structured: JSON or XML files (the schema or structure may change)
* Unstructured: video, audio, logs

**Data Scientist versus Data Engineer**



**Common Languages Used?**

* Scala - this is usually used in once the production is live
* Python - this is usually used in pre-production
* Java
* R

**Common IDEs Used:**

* NTELLIJ
* ECLIPSE
* PYCHARM

**Pipeline and/or Architecture:**

Every pipeline in the industry is composed of 5 to 6 components:

Sources --> Ingestion --> Processing --> Storage --> Analytics --> Visualization

(Sometimes, these steps can go in other order, not just the order)

Sources: The source of the raw or structured data

- can come from IoT devices, like cell phones, thermometer, sensors - anything connected to internet that's measuring information

- can come from REST services (online)

- can come from Clickstream (every time you move your mouse over a webpage, it tracks every millisecond of where you placed your mouse)

- can come from SOAP services

- can come from Databases

- can come from Data Warehouses (essentially, a big database)

Ingestion: This is the process of using tools to read information from the "Sources" to process (or ingest) or transport this information into the data pipeline. Some ingestion tools:

ON PREM TOOLS:

- Kafka (reads information that is in steams or streaming - continuous information without stopping - cellphone sending information to Google every single second, for instance) - MOST COMMONLY USED INGESTION TOOL

- Messaging System (JMS - java messaging system, Rabbit MQ, SQS - simple queue service)

- Sqoop (takes information from relational databases into the pipeline - Oracle, MySQL - these are examples of relational databases)

- Flume (moves logs from one applicaiton that produces logs into Hadoop)

- All the above were designed as tools "on prem" (can run on databases), however these can run on cloud, too, if these are installed on the sever in the cloud - it's a an "unmanaged" service, because, unlike the cloud tools (whereas MS or Amazon is helping to support and run, YOU have to run these)

CLOUD TOOLS:

- AWS - Amazon Web Service (Kinesis - same thing as Kafka, it's just in the cloud)

- Azure - (Event Hub - same thing as Kafka, it's just in the cloud)

- GCP - Google Cloud Platform - (DataFlow - same thing as Kafka, it's just in the cloud)

Processing: Anything that manipulates the data that you are handling in your pipeline...

2 Types of processing in the pipeline:

- "Batch"

Tools for batch processing:

- PIG

- HIVE

- Map Reduce (no longer used in the industry, but important to know about in case it's a legacy technology on a previous project)

- "Streaming" (market is moving more toward streaming data, in fact, moving some production environments that are currently Batch Processing into Streaming Processing)

Tools for streaming processing:

- Flink

- Storm

- Kafka Streams

Tools that handling \*both\* Batch and Streaming Processing:

- Spark (this also handles Machine Learning, too - not just batch and streaming processing) - most common one taught in training, btw, as it handles both

Storage: (Any place you store the information you're handling in the pipeline). There are three types of storage:

1. Data Lakes - these save the information wtihout the schema - without forcing any structure on the data - this means in the data lake, you can save a video, as well as am Excel file, as well as a JSON file, etc. (you don't have to define the structure or type)

- Note - there are different Data Lakes - most common is HDFS. Another one is S3 (made by AWS), another one is Blob (made by Azure), another one is D23 (made by Dell)

2. Data Warehouses - these are like data bases, but larger and distributed (you have multiple computers contributing to the same goal, a "Cluster"). These will all be housing structured data. Here are some types of DWH:

- HIVE (most commonly covered during training) - you query the data warehouse with SQL - reminder, this is ON PREM

- Redshift (AWS tools)

- Teradata

- Azure Data Warehouse

3. Databases - structured data - note, you'd move to a Data Warehouse if you forecast your data to grow exponentially or simply have very large amounts of data (3 petabytes +) for reference (1 bit, 1 byte (8 bits), 1 MB (1024 kbytes), 1 GB (1024 MB), 1 TB (1024 GB), 1 PD (1024 TB).

Types of relational databases:

- MySQL

- Oracle

Types of no-SQL (or non-relational) databases:

- HBASE

- Cassandra

- Mongo

- Elastic Search

Analytics (not covered extremely heavily for Big Data/Hadoop, as it's more part of Data Science, but there are frameworks used to help analyize the data):

- Spark ML

- SCI-KIT Learn

- Numpy

- TENSOR FLOW

Visualization Tools: (all of these are variations of the same)

- Kibana

- Tableau

- PowerBI

- Grafana

- Kibana

**More on Talking about a Pipeline**

The consultant's goal and capability should be to be able to speak to the right pipeline for each project based on the end client's needs, type of data, and which tools interact and work together, again, across the pipeline areas of: Sources, Ingestion, Processing, Storage, Analytics, and Visualization.

Above-listed pipeline areas are part of a general pipeline. However, there can be additional items added:

Hadoop Distributions or "vendors" could be considered - this is the "type of environment" a consultant could be running. There are three major companies or vendors that have pre-packed pipeline materials for Hadoop:

- Cloudera

- Hortonworks

- MapR - these three distributions already have a pre-made "suite" of the necessary pipeline tools already packaged together. Note, these are all ON PREM.

There are three counterparts in the CLOUD that are countering Cloudera, Hortonworks, and MapR. They are:

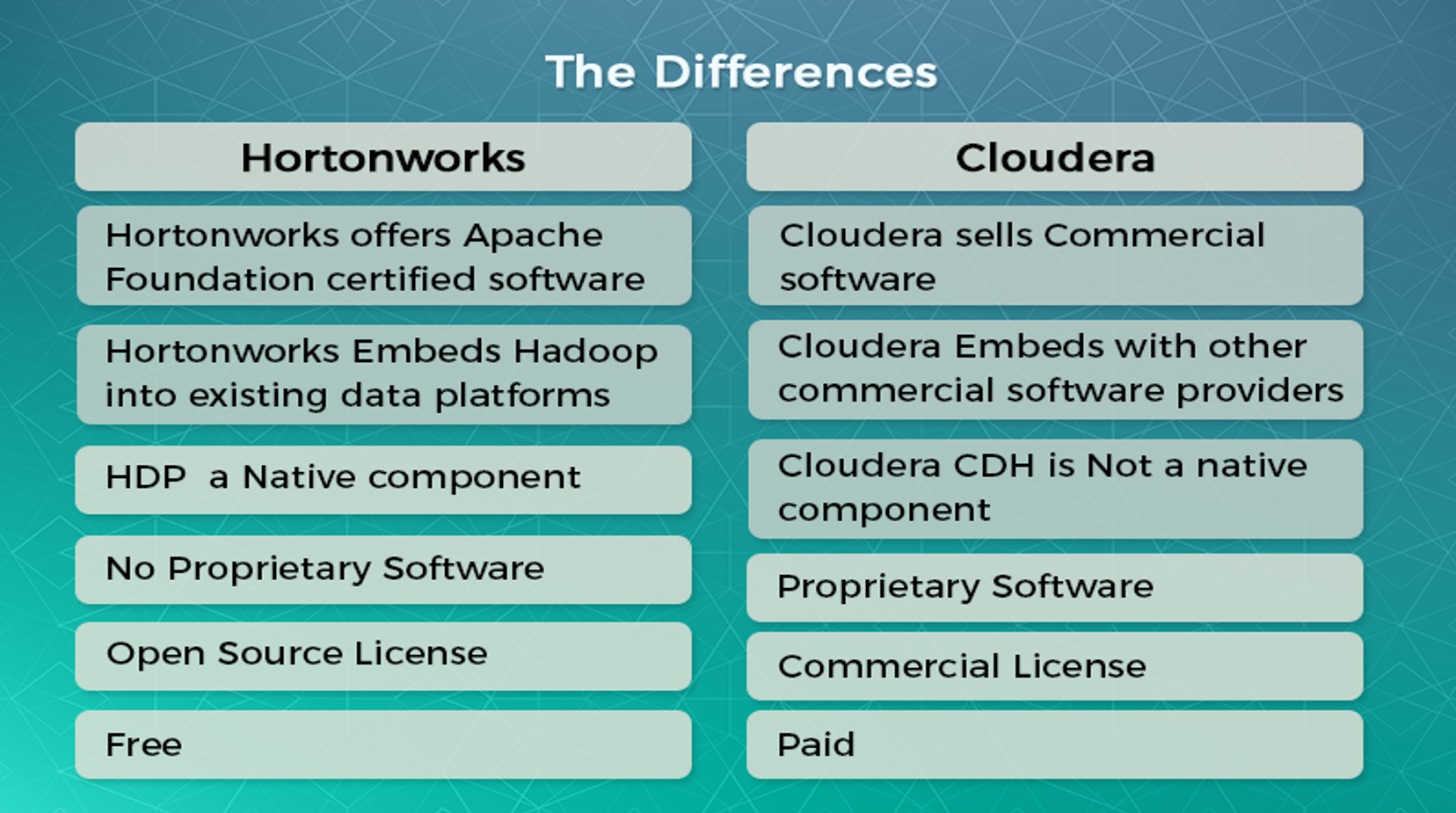
- HDInsights - Azure

- EMR - AWS

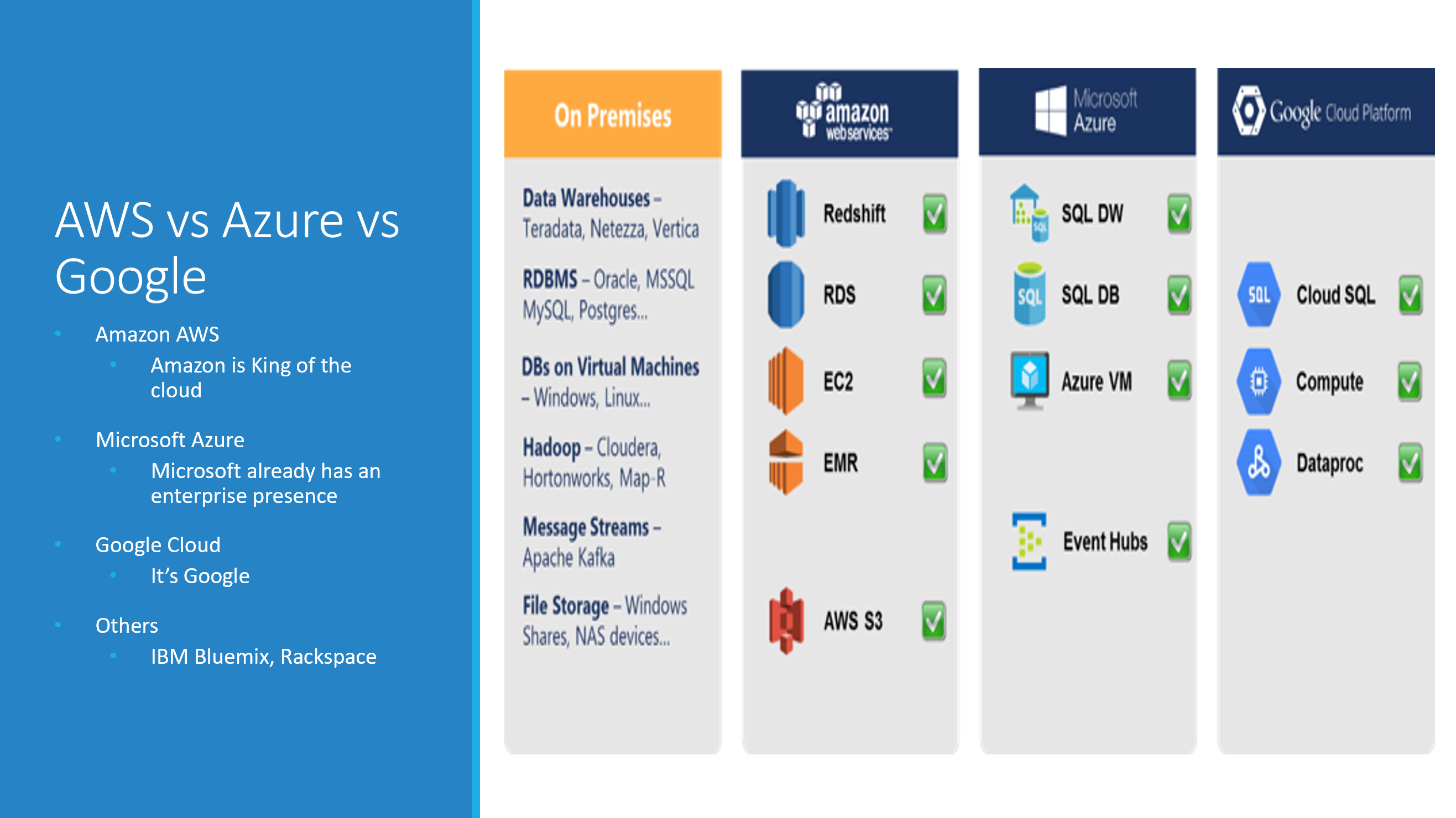
- GCP - Google DataProc

(these cloud versions have the exact same tools as the on prem versions, so if you know the on prem versions, you should be able to talk about the cloud versions)

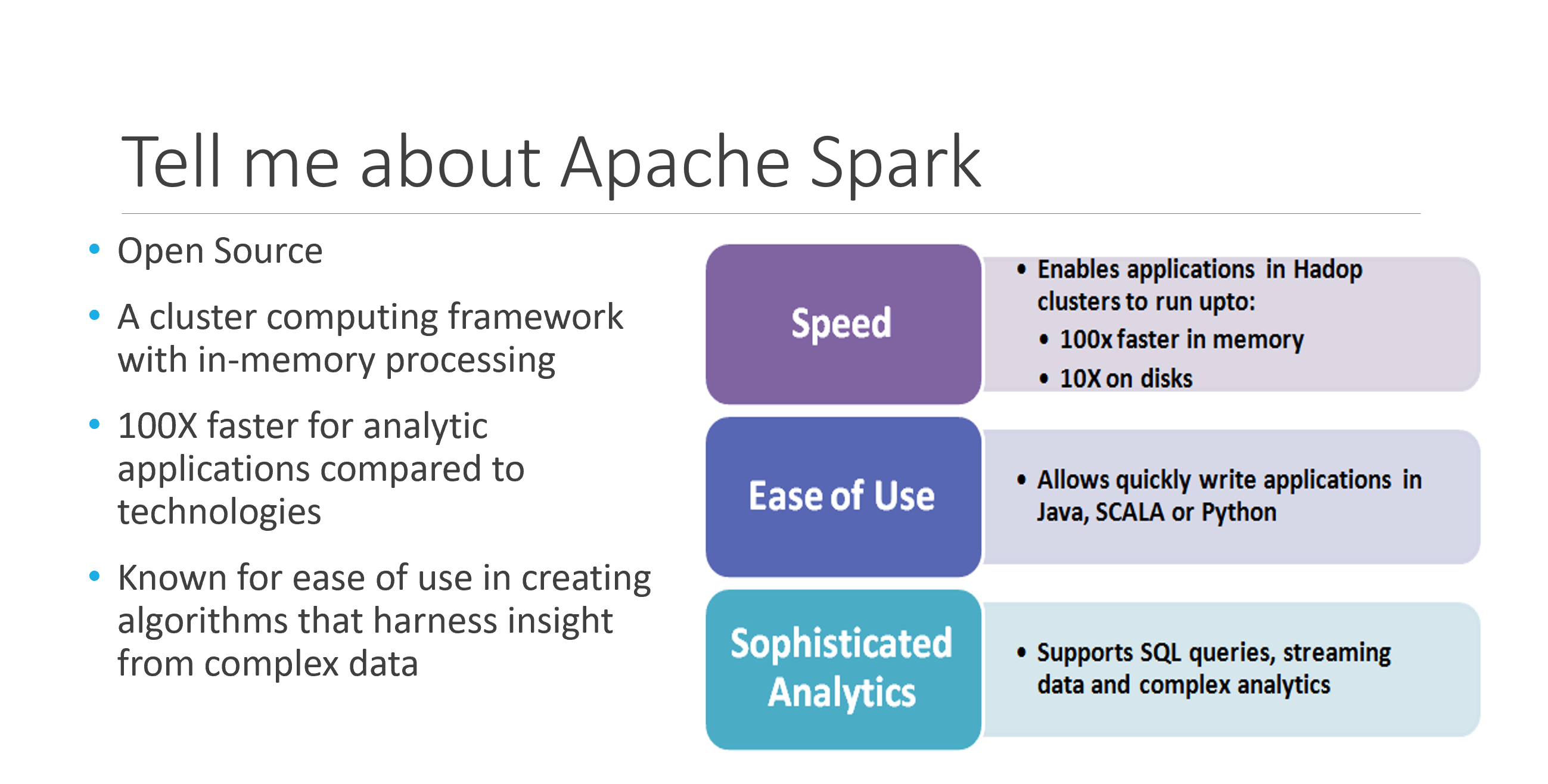
**More on HortonWorks versus Cloudera**



**More on AWS versus Azure versus Google**



**More about Spark**



**Several Other Commonly Recurring Concepts:**

1. Spark - in relation to the actual projects on the marketing profile - consultant must know about Spark in relation to those

- Architecture of Spark

- Internal operation of Spark (how it actually works)

- Spark API

- Coding concepts of Spark

2. Kafka - in relation to the actual projects on the marketing profile - consultant must know about Spark in relation to those

- Architecture of Kafka

- Internal operation of Kafka (how it actually works)

- API

- Coding concepts of Kakfa

3. AWS

- Glue

- RDS (Oracle, Postgres, MariaDB)

- LAMBDA

- ATHENA

- SAGE MAKER

- IAM

- Security Groups

- Centrify

- VPN (Vehicle Private Network)

- AIRFLOW

- DATA PIPELINE

- ECS (Elastic Container Service)

- EBS (Elastic Block Service)

- ELB (Elastic Load Balancer)

- ELK (Elastic Logstash Kibana)

- EMR (Elastic Map Reduce) - Spark, HIVE, OOZIE, SQOOP, GANGLIA, TEZ, KNOW, RANGER

- SWF (Simple Workflow Service)

- LDAP (Lightway Directory Active Protocol)

- KERBEROS (security for the cluster - authentication ticketing systems)

- SSL (encryption)

- SSH

4. DevOps

- ANSIBLE (automation tool for deployment)

- PUPPET (automation tool for deployment)

- CONTAINERS (like virtual machines - there are Dockers, and Kubes) - also, part of the automation

- Cloud Formation (this is an AWS tool) - also, this is part of automation - "Infrastructure as code" (meaning, if I want X amount of servers, for instance, I actually code it)

- Jenkins (continuous integration - and actually setting up the CI server)

- Git, BitBucket (Version Control)

- Jira, Trello, etc. (task assignment and tracking)

5. Spark (deeper dive into the architecture):

-> Master Slave --> (Resource Manager) Yarn, Mesos, or Spark --> Slaves, Slaves, Slaves

6. Spark (deepr dive into the internal operation, or how it works):

- Takes data in operations to break the data in chunks and process it simultaneously, so a large set of data can be processed quickly, at the same time

- After this, there are three major components inside Spark that consultants use:

1 RDDs - Resilient Distributed Data Sets (a collection of immutable elements that can be broken and spread among the nodes inside a cluster to be processed in parallel). These are schema-less

2 Data frames - same concept of RDD, however, they have schema (they look like a table - they're structured)

7. Data sets - like data frames, but strongly typed (you can identify the information or type of data within each columns)

- You do TRANSFORMATIONS and ACTIONS with the RDDs, DF, or DS - keep in mind these are all immutable (this means you can't change them), so when you do transformations (transforming the data you have in a dataframe), you have several options:

1. Map

2. Filter

3. Sort

4. groupByKey

5. etc...

- They need to know which transformation types they used in the past projects, and why

- You then do ACTIONS after the transformations - this takes the output of the dataframe (and transformation), and it sends the output to where you want to send it

1. .collect

2. .save

3. .saveasJson

4. etc.

- Any operation that implies I/O (input/output), it's considered an action

APIs of Spark (composed of 4) - Spark Core (the foundation or blueprint for all the 4 APIs):

1. Spark SQL - a way Spark parallelization information (data sets, data frames, and RDDs), and process, using a quert language - in this case SQL - (consultants use this 100% in training - must know this)

2. Spark Streaming - information that is traveling from one place ot another, constantly - takes care of this data and provides transformations and actions to said data. (Example: Facebook app - sends your GPS location - this location has X/Y coordinates and sends this to an application to a server. There are about 200M users right now in the world sending this information. After this goes through the app server, another app looks for specific information within this info. Then it's saved in a database. Then it's analyzed.

Streamling Pipeline: Facebook --> Data (such as GPS) --> App Server --> Another app looking through the data and filters, transforms, and actions upon the data we want out of the constantly streaming data (SPARK STREAMING takes place here) --> Saved in database --> filtered data is Analyzed

Dstream is an RDD in movement, and the way you get the Dstream is via the pipeline mentioned above.

3. Spark Machine Learning - not heavily asked about in interviews, but good to know this is one of the 4 APIs within the spark core

4. Spark Graph X - not heavily asked about in interviews, but good to know this is one of the 4 APIs within the spark core

**(For Streaming Pipeline) Spark Core and the 4 APIs depend on Spark Session**

Spark Session:

1. It's the library that allows the interaction of your program ("Driver" is the name of the program) with the cluster.

2. If I don't use Spark Session, I can still do maps, filters, and other transformations, but they won't be interacting with the cluster (the multiple computers/servers processing the data simultaneously)

3. Spark Session was released on Spark 2.0 - before this, we have Spark Context - this does the same thing as Spark Session. It's just that Spark Session has more APIs and more optimized

4. To create a Driver - one uses Scala, Python, Java, or R - BD engineers need to know all these

- One must use a tool called SBT (Scala Building Tool) to create the driver

- In the SBT file, you instantiate all the libraries you will be using

- The prefered IDE is INTELLIJ

- Your program (aka the Driver) should be written with the following parameters: a. define Spark Session, b. define Spark SQL, c. if streaming, then we must define Spark Streaming Context - you also must know what's inside each of these parameters

**Kafka** - **in relation to the actual projects on the marketing profile - consultant must know about Spark in relation to these:**

- Architecture of Kafka

- Internal operation of Kafka (how it actually works)

- API

- Coding concepts of Kakfa

Kafka - A messaging system that allows publishing and subscribing ("pub-sub") - delivers messages and transactions every second. Transmits data from one place to another place

1. It's a "broker" messaging system

- PRODUCER --> CLUSTER --> Consumer

a. Producer - a program that produces any information to Kafka - this info in the producer is sent to KAFKA. The producer requires coding, and that means using Java/Scala/Python/C#/C++/.NET

b Cluster - the producer's information is sent to the cluster. The cluster is composed of the multiple servers or computers to simultanesouly handle the data. These multiple computers are called "brokers." Inside the brokers, you have special pipes or channels, called "Topics." The topics can be partitioned (broken up to send information in parallel). Whatever is in one topic in one broker, it will be replicated in another topic in another broker.

c. Consumer - any application or program that reads the TOPIC'S output. Streaming programs are the only ones that can read the KAFKA pipelines topic output - that can be:

- Spark

- Flink

- Storm

2. Question about ZOOKEEPER: What happens if the Broker 1 goes down? We need something to tell the Producer to send the information to Broker 2 or Broker 3 (the still-functional brokers), since the data from Broker 1 was replicated in Brocker 2 and Brocker 3. That's called ZOOKEEPER.

3. Question about OFFSET (a number that ZOOKEEPER Tracks): What happens if the Consumer dies? Say for instance that data of "A, B, C, D, E" is being sent to the Consumer, and the Consumer dies after "E, D" have been sent across from the Topic to the Consumer - how does the consumer know to pick back up from "C"? - OFFSET

a. The OFFSET gets attached to each message

b. How can you ensure only one message is being consumed? How do you avoid duplicates? How do you avoid offsets aren't being repeated? Knowing about the delivery mechanisms in KAFKA helps to avoid this.

- There are 3 known mech

**Additional Common Questions Re: Spark + Kafka:**

Spark Heavy Questions:

1. What is the business case of your pipeline (the problem your company is trying to solve)?

2. What is your architecture of your pipeline(s)?

3. What technology stack have you used, and why?

4. What is your specific contribution within the pipeline?

5. Difference between RDD, data frame, and data set?

6. What is the catalyst optimizer?

7. What is the project tungsten and when do you use it?

8. What's the size of the data?

9. How do you calculate the size of your data?

10. How many nodes are in the cluster?

11. How do you calculate the number of executors and memory inside your cluster?

12. What is data skewness?

13. How do you solve the problem of skewness?

14. What is a broadcast variable?

15. What kind of transformations have you been doing in your data?

16. Difference between groupByKey versus reduceByKey?

17. What is shuffling?

18. How do you decrease shuffling?

19. Common issues with memory in Spark?

20. Difference between coalesce and repartition (or partition)?

21. What is the standard garbarge collector for Spark?

22. How to read a file in Spark?

23. Difference between Spark and MapReduce?

24. What is lazy evaluation?

25. What is a trait?

26. What are case classes and what are they used for?

27. What is the DAG?

28. Define the lineage of a Spark job.

29. How do you submit your job to a Spark cluster?

30. How to create RDDs in Spark?

31. How to convert RDD to Dataframes?

32. What are the partitions in Spark

33. What is an action?

34. What are transformations?

35. What is a pair RDD?

36. What is the Driver?

37. What is an executor?

38. Tell me what are the submissions modes for a Spark job?

39. Type of cluster managers in Spark?

40. What is a sparse vector?

41. What is an accumulator?

42. What are sliding windows?

43. Define caching in Spark.

44. Levels of persistence in Spark?

45. Difference between persistence and cache?

46. What are check points?

47. What is fault tolerance in Spark?

48. Different types of compression for Spark?

49. What is a map, versus a flat map, versus a fold?

50. What is the difference between an input block versus a partition in an RDD?

KAFKA Heavy Questions:

1. What is Kafka?

2. What is a topic?

3. What's a producer? a consumer?

4. What are the brokers?

5. Why do we use the offset?

6. What happens if broker A goes down?

7. What happens if consumer goes does?

8. Explain the role of an offset.

9. Explain the role of Zookeeper and how it interacts with Kafka.

10. What are the partitions in Kafka?

11. Describe your Kafka architecture.

12. Explain the concept of leader and follower.

13. What are ISRs?

14. How do you ensure parallel consumption from Kafka?

15. What's the default retention for Kafka?

16. What is multi-tenancy in Kafka?

17. How do you handle change of schema in your Kafka scheme?

18. Differnce between Confluent and Kafka?

19. What is the log anatomy?

20. How do you tune Kafka?

21. How does Kafka log aggregation?

22. Features of Kafka streams?

23. What are the delivery mechanisms for Kafka?

24. Why is better to code in Java for Kafka?

25. Difference betweem a Broker and a Server?

26. What are serdes?

27. Difference between Flume and Kafka?

28. Difference between Kafka, RabbitMQ, JMS?

29. Explain how you can get exactly one message from Kafka.

30. Why is replication required in Kakfa?

31. Where do you save your offsets?