Question Set A: The Ambari Dashboard

The Ambari Dashboard is intended to provide an overview of the Hadoop cluster. In your opinion:

1. Describe three ways in which the Ambari Dashboard provides you with useful information.

   Metrics, Heatmap and Configuration history

2. Mention other information you would like to have access to on the Dashboard, and when/how you would have use of it.

   - Cluster Configuration Changes:
     History of configuration changes made to the cluster components, along with who made the changes and when. This helps maintain an audit trail and ensures accountability for configuration modifications.
   - Resource Allocation and Queues:
     Information on resource queues, capacity, and resource allocation policies within YARN. This is especially useful for cluster administrators to ensure fair resource allocation among different applications and user groups.
   - Backup and Data Recovery Status:
     Information about the status of cluster backups, snapshots, and data recovery processes. This is essential for disaster recovery planning and ensuring data availability.

Question Set B: Services

1. What is a “service”?

   A "service" refers to a specific software component or application that serves a particular purpose within the larger system. These services are typically designed to work together to provide a comprehensive platform for processing and managing large datasets.

2. What services (names only) are available in this cluster?

   HDFS, YARN, MapReduce2, Hive, HBase, Oozie, ZooKeeper, Storm, Infra Solr, Atlas, Kafka, Knox, Ranger, Spark2, Zeppelin Note, Data Analytic, Druid, Superset.

3. What are the services used for? (Choose three services; search for information in the lecture slides and on the Internet and relate to what you see in Ambari in two sentences for each service.)

- Hive:
  - Role: Hive is a data warehousing and SQL-like query language service for Hadoop. It allows users to query, analyze, and manage large datasets stored in Hadoop using SQL-like queries.
  - In Ambari: In Ambari, you can monitor Hive's health and performance, view its configuration settings, and restart or stop the Hive service if necessary. You can also access Hive's web-based user interface to create and manage tables, write SQL queries, and execute them on the Hadoop cluster.

- HBase:
  - Role: HBase is a NoSQL database service built on top of Hadoop that provides real-time read and write access to large datasets. It is suitable for use cases requiring low-latency data access and is often used for time-series data or applications requiring random access to large amounts of data.
  - In Ambari: In Ambari, you can monitor HBase's health and performance, view its configuration settings, and make adjustments to its configuration as needed. You can also manage HBase tables and regions, set up replication, and ensure that the HBase service is running efficiently.

- Spark2:
  - Role: Spark is a distributed data processing and analytics framework. Spark2, as mentioned in Ambari, likely refers to a specific version of Spark. Spark allows users to perform batch processing, interactive queries, and stream processing, making it a versatile tool for data analysis.
  - In Ambari: In Ambari, you can monitor Spark2's health and performance, view its configuration settings, and manage Spark applications and jobs. You can also adjust Spark2 configurations, allocate resources, and ensure that Spark2 is running smoothly to process data efficiently within the Hadoop cluster.

4. What information is available for several, most, or all, services?

- Service Status: Information about whether the service is running, stopped, or in a particular state (e.g., starting, stopping, or restarting).
- Health Metrics: Health indicators or metrics that show the overall health of the service. These metrics are often color-coded to provide a quick visual assessment of the service's status.
- Configuration Settings: Access to configuration files and settings for the service. Administrators can view, modify, or override configuration parameters to tailor the service to their specific requirements.
- Service Logs: Access to service logs and log files, which can be helpful for troubleshooting and diagnosing issues within the service.
- Resource Utilization: Metrics related to resource utilization, such as CPU usage, memory usage, and network usage. These metrics help administrators monitor the service's resource consumption and performance.

- Component Status: Information about the individual components or daemons associated with the service. This includes details on whether component instances are running, their statuses, and resource usage.
- Actions and Operations: The ability to perform actions on the service, such as starting, stopping, restarting, or scaling components. Additionally, you can initiate common operations like rolling restarts or service checks.
- Service Dependencies: Information about dependencies between the service and other services within the cluster. This helps administrators understand how different services interact and rely on each other.
- Security and Access Control: Settings related to security features, access control, and authentication methods used by the service. This is essential for maintaining the security of the Hadoop cluster.

5. Is there some information for some service that you would like to have available already on the Dashboard?

   - Recommendations and Auto-Scaling:
   Incorporating intelligent recommendations for cluster optimizations, such as resource allocation adjustments, configuration tuning, or scaling actions based on historical usage and performance data.

- Topology and Network Visualization:
  Providing a visual representation of the cluster's network topology, including network traffic, data flow, and component interactions. This can help administrators troubleshoot network-related issues more effectively.

- Custom Alerts and Notifications:
  Having the ability to set up custom alerts and notifications based on specific conditions or events within the cluster. This would allow administrators to proactively respond to critical situations or performance anomalies.

- Job History and Analytics: Access to more detailed historical information and analytics regarding job executions. This could include job-level performance statistics, historical job trends, and the ability to visualize job dependencies.

Question Set C: Service Actions

1. What do actions “Start” and “Stop” do?

   Start: The "Start" action initiates the process of starting a service or component that is currently in a stopped or inactive state. When you start a service or component, Ambari sends commands to the cluster to launch the necessary daemons or processes associated with that service.
   Starting a service makes it available for use, allowing it to perform its intended functions within the Hadoop ecosystem. For example, starting the HDFS service enables the

storage and retrieval of data, while starting the YARN service allows for resource management and job execution.
Stop: The "Stop" action initiates the process of stopping a service or component that is currently running. When you stop a service, Ambari sends commands to gracefully shut down the service's daemons or processes.
Stopping a service halts its operations and releases the associated cluster resources. This can be useful when performing maintenance, making configuration changes, or temporarily disabling a service that is not currently needed.

2. What do actions “Turn On Maintenance Mode” and “Turn Off Maintenance Mode” do?

   Turn On Maintenance Mode: When you select the "Turn On Maintenance Mode" action for a specific service or component, you are indicating that you want to put that service or component into a maintenance state.
   Maintenance mode is typically used when you need to perform maintenance tasks, such as software updates, configuration changes, or hardware repairs on the service or component. Activating maintenance mode allows you to stop the service temporarily without triggering alerts or causing unnecessary disruptions to the cluster's operation.
   While a service or component is in maintenance mode, it will not actively process data or perform its normal functions. However, it can be safely worked on, updated, or reconfigured.
   Turn Off Maintenance Mode: The "Turn Off Maintenance Mode" action is used to reverse the maintenance state of a service or component that has been previously put into maintenance mode.
   When you turn off maintenance mode, the service or component returns to its regular operational state. It starts processing data and fulfilling its intended functions within the cluster.
   Turning off maintenance mode essentially signifies that the maintenance tasks have been completed, and the service or component is ready to resume its normal operations.

3. What does the action “Run Service Check” do?

   Check the state and status of the service.

Question Set D: Host Management

1. What is a “host”?

   In the context of distributed computing and Hadoop clusters, a "host" refers to a physical or virtual machine (computer) that is part of the cluster's infrastructure. Each host typically runs an operating system and serves as a node within the cluster. Hosts are fundamental building blocks of a distributed computing environment like Hadoop.

2. How many hosts are there in this cluster?

   Only one, sandbox-hdp.horton

3. What information can you see about each host?

   Name, IP address, Rack, Corse, RAM, Disk Usage, Load Avg, Versions, Components.

4. What does "Rack" mean? Why is that important information; when would you use this information?

   A Rack is essentially a collection of closely interconnected hosts that are physically located in the same area, typically within the same row or section of a data center. All hosts within a rack share a common network switch and often have faster and more reliable network connectivity with each other compared to hosts in different racks.

   Why It's Important Information: Rack information is important for several reasons:

   - Data Locality: In Hadoop clusters, data locality is crucial for efficient data processing. When a Hadoop job is submitted, the goal is to execute tasks on the same rack where the data is stored. This reduces network overhead and improves job performance because data doesn't need to traverse the network extensively.

   - Fault Tolerance: Hadoop is designed for fault tolerance. By having data replicas on different racks, the cluster can withstand rack-level failures without data loss. This is a critical aspect of ensuring data reliability and availability.

   - Network Load Balancing: Rack information is useful for network load balancing. When distributing tasks across the cluster, the Hadoop scheduler can aim to balance the load on different racks, preventing network congestion on a single rack.

   - Hardware Maintenance: Knowing the rack where a host is located is essential for hardware maintenance and replacement. When a host experiences a hardware failure or requires maintenance, administrators can identify the rack and replace the hardware accordingly.

   When You Would Use This Information: would use rack information in various scenarios, including:

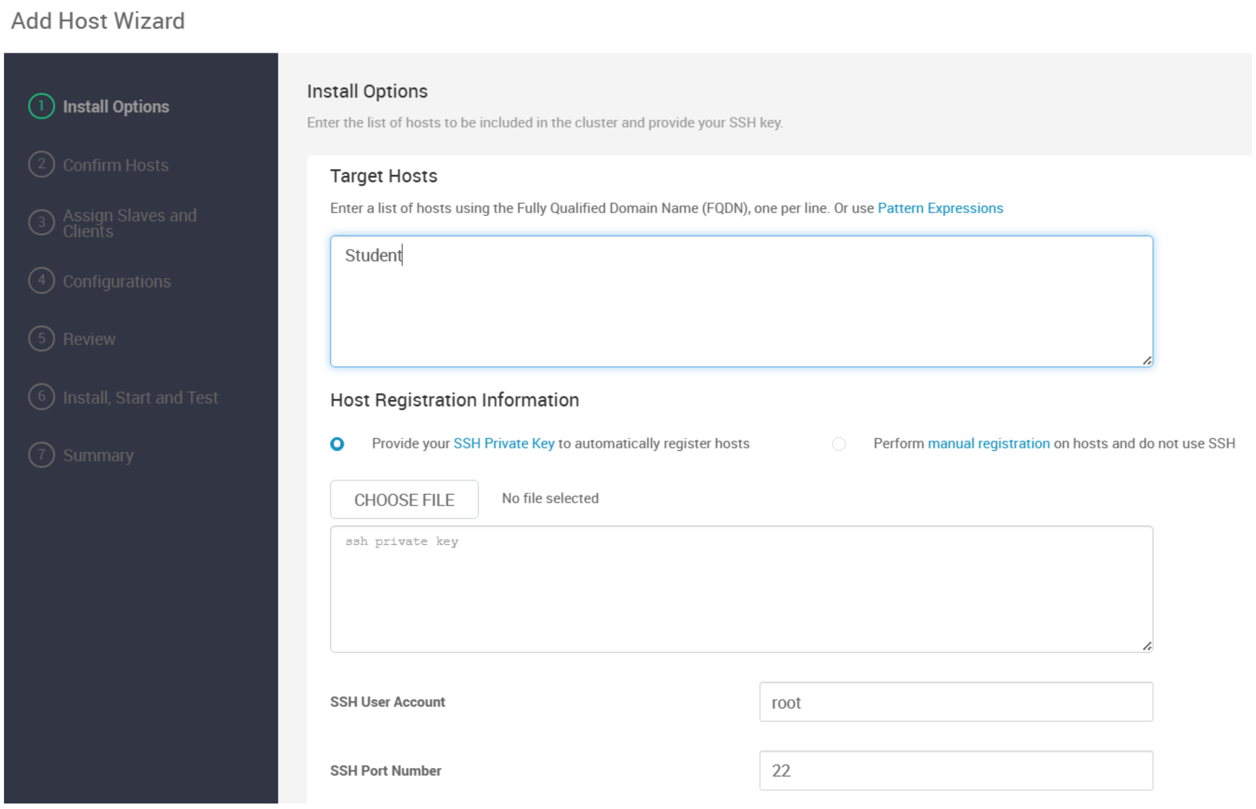
   - Job Scheduling: When configuring job schedulers like YARN, you can set preferences to prioritize data locality by specifying that tasks should be scheduled on the same rack as the data they need to process.

   - Data Replication: When configuring HDFS, you can control data replication policies to ensure that replicas are placed on different racks for fault tolerance.

   - Troubleshooting: When diagnosing network or performance issues within the cluster, knowing the rack can help pinpoint whether the issue is rack-specific or affecting multiple racks.

- Maintenance: When performing maintenance tasks or replacing hardware, having rack information ensures that you address the correct set of hosts.

5. Describe briefly what happens when you (try to) add a host. Speculate what would happen if you actually had another computer connected and added this.



Add Host Wizard

Install Options

Enter the list of hosts to be included in the cluster and provide your SSH key.

Target Hosts

Enter a list of hosts using the Fully Qualified Domain Name (FQDN), one per line. Or use Pattern Expressions

Student

Host Registration Information

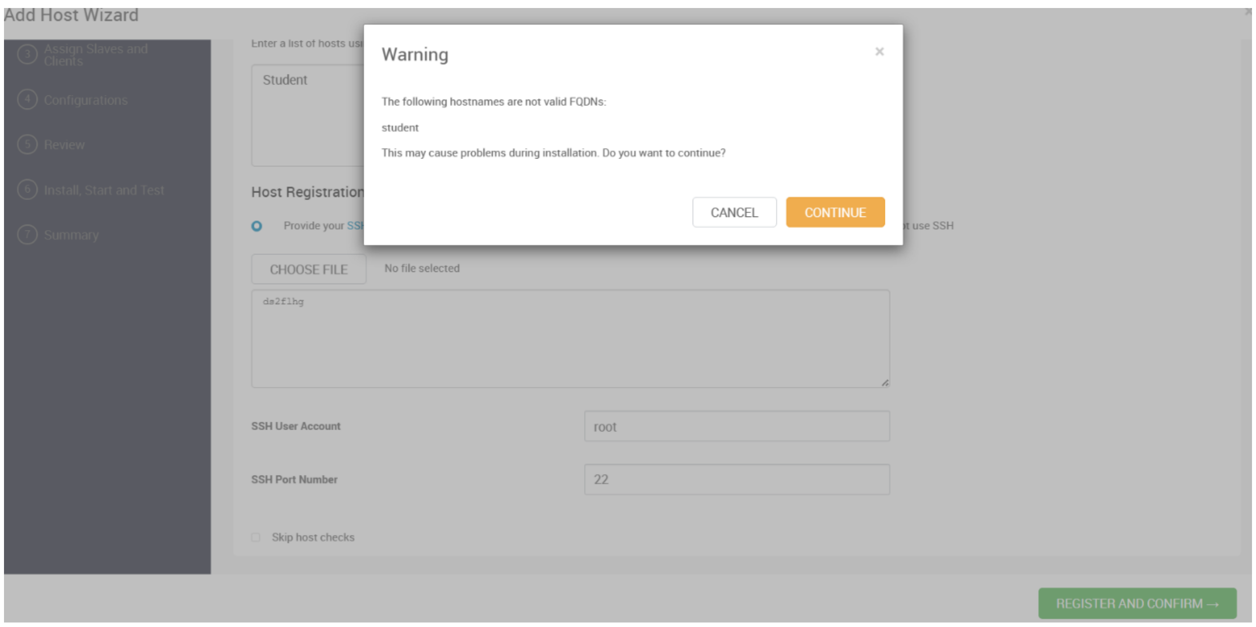
● Provide your SSH Private Key to automatically register hosts ○ Perform manual registration on hosts and do not use SSH

CHOOSE FILE No file selected

ssh private key

SSH User Account: root

SSH Port Number: 22



Add Host Wizard

Warning

The following hostnames are not valid FQDNs:

student

This may cause problems during installation. Do you want to continue?

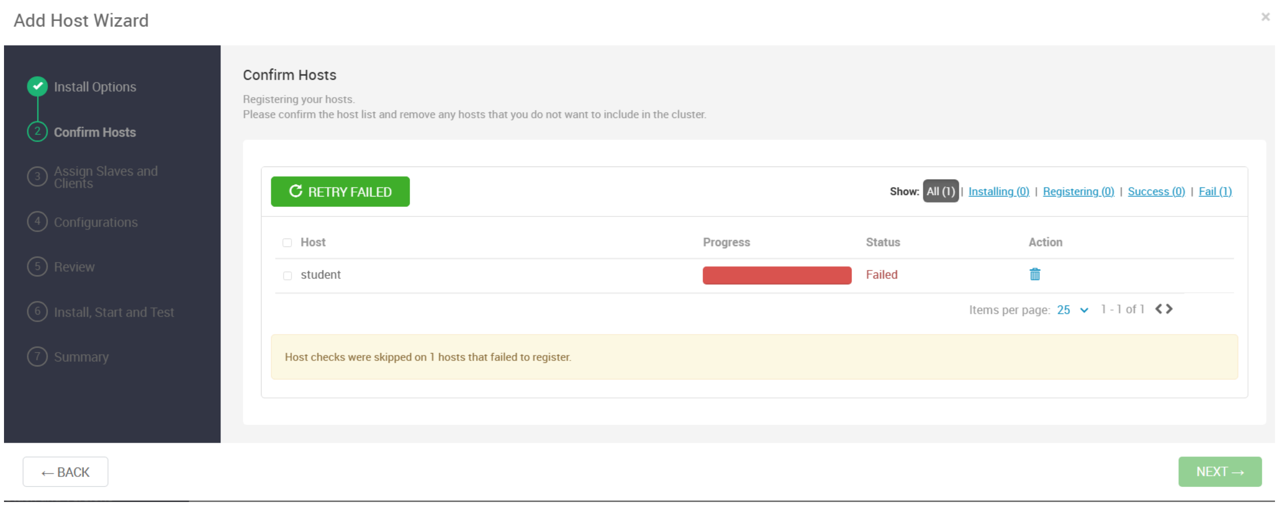
Student

da2f1hg

SSH User Account: root

SSH Port Number: 22

Skip host checks



Add Host Wizard

Confirm Hosts

Registering your hosts.
Please confirm the host list and remove any hosts that you do not want to include in the cluster.

Host	Progress	Status	Action
student	🟥	Failed	🗑

Items per page: 25 1 - 1 of 1

Host checks were skipped on 1 hosts that failed to register.

I have failed to add a host. I think when trying to add another host while another computer is connected I think the system would attempt to connect to the computer using SSH since it was asking about SSh private key as in the first photo. Then if the credentials are correct and the computer is accessible, then host would be added successfully, then the status would then show as successful, not failed as I got in last photo.

Question Set E: HDFS

1. Describe HDFS briefly in your own words; in what way is HDFS a cornerstone in Hadoop? (Two to three sentences.)

   HDFS, or the Hadoop Distributed File System, is a scalable and fault-tolerant file storage system designed to store large volumes of data across multiple machines in a Hadoop cluster

2. Does HDFS seem to be running well on this cluster? Why do you think so?

   It is running but it is taking too much time, it could be for several reasons:
   a. Garbage Collection: Java Virtual Machine (JVM) garbage collection processes can sometimes cause pauses, especially if the heap size is not optimally configured, affecting performance.
   b. Large Number of Small Files: HDFS is optimized for a small number of large files rather than a large number of small files. Managing a large number of small files can create overhead for the NameNode, as it has to keep track of the metadata for each file.

3. What does NAMENODE and SNAMENODE mean? Are both started?

   NameNode: is the centerpiece of an HDFS file system. It manages the file system namespace and regulates access to files by clients. Essentially, it keeps track of the directory tree of all files in the system, maintains the metadata for all HDFS files and directories (such as the file name, permissions, and the list of blocks and their locations for each file), and manages the opening, closing, and renaming of files or directories. However, it does not store the actual data of these files; the data itself is stored in another type of node called the DataNode. The NameNode is a critical part of the HDFS architecture because if it fails, the file system cannot be accessed.

Secondary NameNode: The term "Secondary NameNode" is somewhat misleading because it doesn't serve as a backup NameNode in the traditional sense of a secondary system taking over in case the primary fails. Instead, the Secondary NameNode performs housekeeping tasks for the NameNode, such as periodically merging the fsimage (the file system metadata snapshot) with the edit log (a record of changes made to the metadata) to prevent the edit log from becoming too large. The processed data is then sent back to the NameNode, which helps in improving the startup time of the NameNode and ensures that the system has a relatively up-to-date snapshot of the metadata. The Secondary NameNode does not serve as a failover option for the NameNode.

Got only NAMENODE started but not SNAMENODE.

4. Are any DATANODEs running?

   Yes

5. Why are there only one of each kind of NODE (NAMENODE, SNAMENODE, DATANODE)?

   In a typical Hadoop HDFS setup, the architecture is designed to include one NameNode, an optional Secondary NameNode, and multiple DataNodes. This design is based on the original architecture of HDFS, where each type of node has a specific role:

   NameNode: There is traditionally only one NameNode in an HDFS cluster because it acts as the master server managing the namespace of the file system and storing all the metadata for the files and directories. The metadata includes information like the file names, the directory structure, permissions, and the locations of each file's data blocks across the DataNodes. Having a single NameNode simplifies the architecture and ensures a consistent view of the file system. However, this also introduces a single point of failure, which modern Hadoop clusters address through High Availability configurations that use multiple NameNodes in an active/passive setup to provide redundancy.

   Secondary NameNode: The role of the Secondary NameNode is often misunderstood. It does not serve as a backup NameNode but rather performs housekeeping tasks for the NameNode, such as merging the edit log with the file system image (fsimage). This process helps to keep the size of the edit log manageable and reduces the startup time for the NameNode. There is typically only one Secondary NameNode because its role is not to provide fault tolerance or load balancing but to assist the primary NameNode in maintaining the health of the metadata.

DataNodes: Unlike the NameNode and Secondary NameNode, there are usually multiple DataNodes in a Hadoop cluster. DataNodes are responsible for storing the actual data in HDFS. Each DataNode stores a portion of the file system's data as data blocks, and each block is replicated across multiple DataNodes based on the replication factor (usually three by default) to ensure data reliability and availability. The number of DataNodes can scale horizontally to increase the capacity and processing power of the cluster.

6. Describe briefly how you can configure each kind of NODE.
Configuring each type of node in an HDFS cluster involves setting various parameters in Hadoop's configuration files. These files are typically XML files where you can specify properties that control the behavior of the NameNode, Secondary NameNode, and DataNodes. The main configuration files involved are core-site.xml, hdfs-site.xml, and yarn-site.xml (for YARN-related configurations). Here's a brief overview of how to configure each type of node:

- NameNode Configuration

hdfs-site.xml: This file contains configurations specific to HDFS. For the NameNode, you might configure: dfs.namenode.name.dir: Specifies the directory where the NameNode stores its metadata and edit logs. It's crucial to have this on reliable, high-performance storage.
dfs.replication: Sets the default block replication for HDFS files. Although this affects the entire filesystem, it's a critical parameter that the NameNode enforces.
core-site.xml: For the NameNode and the HDFS, you might set:

fs.defaultFS: Specifies the URI of the HDFS filesystem. This is where the NameNode URI is defined.

- Secondary NameNode Configuration

hdfs-site.xml: Like the NameNode, the Secondary NameNode uses this file for configurations like:
dfs.namenode.secondary.http-address: Defines the address where the Secondary NameNode runs its web UI, which can be used for monitoring.
dfs.namenode.checkpoint.dir: Specifies the directory where the Secondary NameNode stores checkpoint images of the namespace.

- DataNode Configuration

hdfs-site.xml: For DataNodes, important settings include:
dfs.datanode.data.dir: Defines the directories on the local filesystem where the DataNode stores its data blocks. These should be on separate drives or partitions for better performance and reliability.
dfs.datanode.address, dfs.datanode.http.address, and dfs.datanode.https.address: Configure the address and ports for DataNode services.

Question Set F: Alerts

1. What is an “alert”?

In a Hadoop environment, an "alert" typically refers to a notification generated by monitoring and management tools within the ecosystem to inform administrators about issues, changes, or noteworthy events occurring within the Hadoop cluster. These alerts can cover a wide range of aspects, from hardware and software health to performance metrics and security concerns.

2. What alerts have been issued during your own lab time? Explain these (very briefly). Some of them related to some clusters I have checking and studying them, and some of alerts related to other services like "Ranger Admin Password check", "Spark2 history server".
3. Why are some alerts marked with status "None"? (Why would you have to be alerted that there is no alert?)

   Indicating that there is nothing wrong, the system check or monitoring process has been completed successfully and that no issues were found.

4. Explain briefly how serious the "NameNode" and "HDFS" alerts would be, if the status was anything else than "None". (Select three alerts and write a sentence each.)

   Unknown: This refers to events whose severity cannot be determined or classified. It might indicate an unexpected or anomalous condition that the logging system cannot categorize.

   Critical: Critical severity level is events that indicate a severe problem or failure within the Hadoop system. These issues often require immediate attention and resolution to ensure the stability and proper functioning of the system.

   Warning: that indicates potential issues or conditions that could lead to problems if not addressed or resolved. It requires attention to prevent potential disruptions or failures in the system. While not as severe as critical issues are.

Question Set G: Cluster Management

1. Characterize the difference between "Cluster Administration" (i.e. all the previous sections) and "Cluster Management".

Cluster management = handling changes of the cluster, upgrade Hadoop/service versions, singular actions made.

Cluster admin = handles changes in the cluster, the daily monitoring and supervision, and starting/stopping services.

Question Set H: Cluster Information

1. Characterize in one or two sentences what you see.

   I see two sections:

Cluster name: Sandbox
And the second section is Cluster Blueprint.

Question Set I: Hadoop Versions

1. Describe in one to three sentences the type of information a cluster manager must understand.

   The version of the services.

Question Set J: Remote Clusters

1. What do you think a remote cluster is?

   A “remote cluster" refers to a Hadoop cluster that is in a different data center or geographic location from the primary cluster or the user accessing it. Remote clusters are often part of a larger, distributed Hadoop deployment strategy, allowing organizations to manage and process data across multiple sites or to leverage cloud resources alongside on-premises infrastructure.

Question Set K: Service Accounts

1. What is it that we see?

   Various service users and groups.

Question Set L: Authorization and Users

1. What information and actions are available in the Ambari user interface, depending on the “User Access” setting? (List three things per user, that are different from some other user.)

   Ambari Admin
   - Full Access to All Clusters: Can view and manage all clusters managed by Ambari, including creating new clusters, adding, or removing services, and configuring all aspects of the cluster.
   - Manage Users and Groups: Has the authority to create, modify, and delete users and groups within Ambari, and assign roles to them.
   - Access to All Settings: Can access and modify all Ambari settings, including Ambari Views configuration, service configurations, and advanced settings like LDAP integration.

   View User

- View-Only Access: Can only view cluster and service information, including metrics, configurations, and alerts, but cannot make any changes.
- Limited Scope: Access is restricted to viewing data for the clusters and services they have been explicitly granted access to.
- No Administrative Capabilities: Cannot manage users, services, or cluster settings, and does not have access to Ambari administrative functions or settings.

Service Administrator

- Limited-Service Management: Can manage specific services assigned by the Ambari Admin, such as starting, stopping, and configuring those services, but cannot modify other services.
- View Specific Service Metrics: Has access to metrics and configurations related to their specific services but cannot view information about other services in the cluster.
- Limited Alert Management: Can view and respond to alerts related to their specific services but does not have access to cluster-wide alerts or the ability to configure alert definitions.

2. What kind of work role do you think this user is intended for?

Ambari Admin
Intended Work Role: IT Infrastructure Manager or Hadoop Administrator.

The Ambari Admin role is designed for users responsible for the overall management and operation of the Hadoop ecosystem. This includes IT infrastructure managers, senior Hadoop administrators, or system architects who need to have overarching control over the cluster management, security settings, user management, and integration with other systems. They are responsible for the health, performance, and security of the Hadoop cluster.

View User
Intended Work Role: Data Analyst or Business User

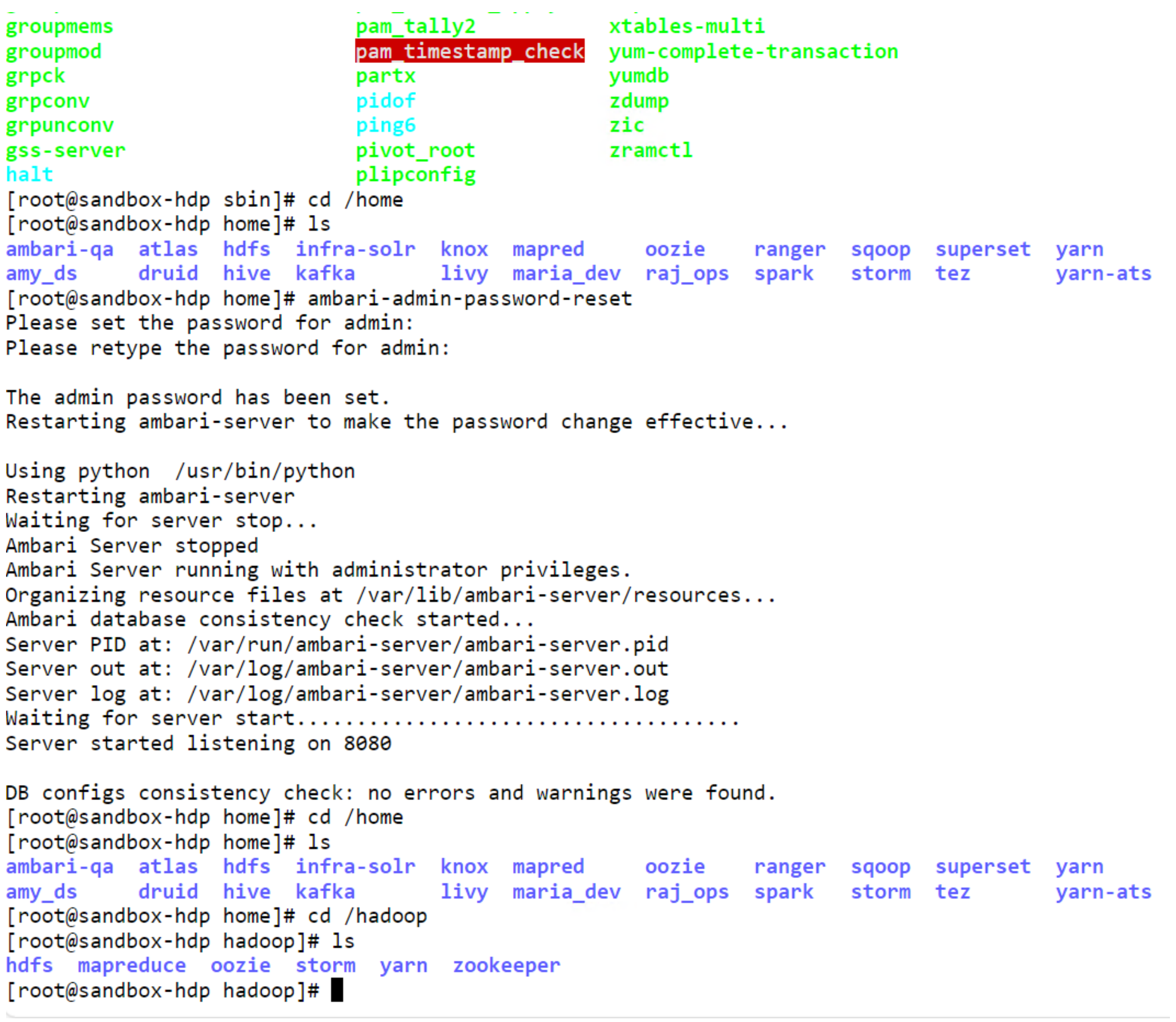
The View User role is designed for users who need to access information within the Hadoop ecosystem for analysis, reporting, or business intelligence purposes but do not need to modify configurations or manage the cluster. This could include data analysts, business users, or other stakeholders who rely on the data stored in Hadoop for decision-making, reporting, or analysis but do not participate in the operational aspects of the cluster management.

Service Administrator
Intended Work Role: Service Specialist or Application Manager

The Service Administrator role is tailored for users who specialize in specific Hadoop services or applications within the cluster, such as HDFS, YARN, or HBase administrators. These individuals are responsible for the configuration, maintenance, and performance optimization of specific services. They might be application managers or technical specialists focused on certain aspects of the Hadoop ecosystem, ensuring that the services they are responsible for are properly configured and optimized.

Question Set M: Cluster Management In Linux

1. Describe how terms and concepts, Hadoop services, Ambari users, etc. that you have met earlier in these instructions, seem to relate to folders in the file structure.



```
groupmems                    pam_tally2           xtables-multi
groupmod                     pam_timestamp_check  yum-complete-transaction
grpck                        partx                yumdb
grpconv                      pidof                zdump
grpunconv                    ping6                zic
gss-server                   pivot_root           zramctl
halt                         plipconfig
[root@sandbox-hdp sbin]# cd /home
[root@sandbox-hdp home]# ls
ambari-qa  atlas  hdfs  infra-solr  knox  mapred     oozie    ranger  sqoop  superset  yarn
amy_ds     druid  hive  kafka       livy  maria_dev  raj_ops  spark   storm  tez       yarn-ats
[root@sandbox-hdp home]# ambari-admin-password-reset
Please set the password for admin:
Please retype the password for admin:

The admin password has been set.
Restarting ambari-server to make the password change effective...

Using python  /usr/bin/python
Restarting ambari-server
Waiting for server stop...
Ambari Server stopped
Ambari Server running with administrator privileges.
Organizing resource files at /var/lib/ambari-server/resources...
Ambari database consistency check started...
Server PID at: /var/run/ambari-server/ambari-server.pid
Server out at: /var/log/ambari-server/ambari-server.out
Server log at: /var/log/ambari-server/ambari-server.log
Waiting for server start....................................
Server started listening on 8080

DB configs consistency check: no errors and warnings were found.
[root@sandbox-hdp home]# cd /home
[root@sandbox-hdp home]# ls
ambari-qa  atlas  hdfs  infra-solr  knox  mapred     oozie    ranger  sqoop  superset  yarn
amy_ds     druid  hive  kafka       livy  maria_dev  raj_ops  spark   storm  tez       yarn-ats
[root@sandbox-hdp home]# cd /hadoop
[root@sandbox-hdp hadoop]# ls
hdfs  mapreduce  oozie  storm  yarn  zookeeper
[root@sandbox-hdp hadoop]# █
```

On Linux (Native Access):

- File System Integration:

Hadoop integrates deeply with the Linux file system, using it for storage directly. HDFS can be mounted on Linux, allowing for seamless interaction.

- Performance:

Native performance is generally better because Hadoop is designed to run directly on Linux. There’s no virtualization layer that could potentially introduce overhead.
Environment Consistency:

A native Linux environment ensures that the services run in the environment they were designed for, which can mean fewer surprises in terms of compatibility and performance.

- Service Management:

Hadoop services are managed through system init tools like Systemd or SysVinit. They are directly installed and run as system services.

- User Management:

Users are managed by the Linux operating system's user management system, which can be integrated with Hadoop’s own security model.

On Windows via Docker (Containerized Access):
- File System Integration:

Docker containers use a layered file system that is abstracted from the host. HDFS data would be stored within the container or in a Docker volume, not directly on the host file system.

- Performance:

Docker on Windows might run on a Linux virtual machine (e.g., through WSL2) or use Windows-native containers. This abstraction can introduce performance overhead compared to a native Linux environment.

- Environment Consistency:

Docker containers encapsulate the environment, which means the Hadoop services run within a Linux environment regardless of the host OS. This ensures consistency across different developers' machines and production environments.

- Service Management:

Hadoop services in Docker are managed within the container context, often with container orchestration tools like Docker Compose or Kubernetes if scaled out.

- User Management:

User management within Docker can be complex. Users are managed within the container and may not align with the host's user system. For Ambari, users are managed by Ambari itself.

Implications for Hadoop Services and Users:

- Presentation of Services:

In a Linux environment, services are accessed and managed more traditionally, whereas in Docker on Windows, services are interacted with through the Docker CLI or other Docker management tools.
Ambari's web UI should look and function the same in both environments, as it is accessed through a web browser.

- Configuration:

Configuration might be more straightforward on Linux, where files can be edited directly. In Docker, configurations might need to be passed as environment variables or configured through Docker-specific files like Dockerfile or docker-compose.yml.
Networking:

Networking in Docker is managed differently, with services within the same Docker network communicating easily, while communication with the host or external network can require additional configuration.

- Data Persistence:

On Linux, data persistence is managed as part of the regular file system, while in Docker, you must manage data persistence through volumes to ensure data is not lost when containers are destroyed.

- Access Control:

On Linux, access control can be managed through traditional file permissions and user groups. In Docker, access control can be more complex because it's handled at the container level and might not map directly to host-level users and groups.
In conclusion, the choice between using Hadoop on Linux directly or through Docker on Windows with Ambari depends on the specific needs, infrastructure, and the environment where Hadoop is being deployed. Each method has its benefits and trade-offs regarding ease of setup, management, performance, and consistency.

2. Why do you think, is there a specific command to reset the password for the Ambari user "admin"?

Security and avoiding mistakes.

Question Set N: The hdfs Command

1. How is it possible to have two file systems simultaneously?

   It's possible to have two file systems simultaneously because a computer can run multiple file systems, each managed independently. For example, Linux can interact with its native file system and HDFS (Hadoop Distributed File System) concurrently.

2. The hdfs command:

   a. In Powershell and in Linux/Unix bash, the tab key completes file and folder names. Why does the tab key not help with HDFS files and folders?

   The hdfs command:
   The tab key doesn't autocomplete HDFS files and folders because HDFS commands run on a client that interacts with the HDFS server, and the shell's autocomplete feature doesn't have access to the HDFS namespace.

   b. Why does not the tilde sign ~ take you to your HDFS home directory?

   The tilde sign ~ doesn't take you to your HDFS home directory because the shell interprets it to mean the home directory on the local file system, not the HDFS.

   c. Why is it not possible to use relative paths in the HDFS file system?

   Relative paths are not used in HDFS because HDFS is designed for high fault tolerance and data availability across a cluster, and relative paths could be ambiguous across multiple nodes.

   d. Why are there no hdfs dfs -pwd or hdfs dfs -cd commands, corresponding to the pwd and cd commands in Linux/Unix?

   There are no hdfs dfs -pwd or hdfs dfs -cd commands because HDFS does not maintain a stateful connection with the current working directory as a local file system does. HDFS commands are stateless; each command must specify the full path in the HDFS namespace.

Question Set O: The WordCount.java Program

1. Which classes are defined in the WordCount.java file?

   Classes defined in WordCount.java:

   WordCount
   TokenizerMapper
   IntSumReducer

2. What does the main() function do?
   Main function:

   Initializes job configuration.
   Sets job name, mapper, combiner, reducer, output key, and value classes.
   Defines input/output paths.
   Starts job processing.

3. Which linesMap phase of the MapReduce algorithm? 5 perform the
a. List these lines.

```
22-27: public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
          StringTokenizer itr = new StringTokenizer(value.toString());
          while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            context.write(word, one);
          }
        }
```

   b. Describe, in natural language, what these lines do; what does the program add as key, what does the program add as value; where is the key-value-pair added to the MapReduce framework?

      Description in natural language:
      These lines define the map function of the TokenizerMapper class, which takes a line of text (value) as input and breaks it into words using a StringTokenizer. For each word tokenized, it sets the word variable to that word and writes a key-value pair to the context, with the word as the key and a numerical one as the value. This prepares the data for the shuffle and sort phase that precedes the reduce phase in the MapReduce framework.

4.Which lines perform the Reduce phase of the MapReduce algorithm?

```
36-44: public void reduce(Text key, Iterable<IntWritable> values,
                 Context context
                 ) throws IOException, InterruptedException {
   int sum = 0;
   for (IntWritable val : values) {
    sum += val.get();
   }
   result.set(sum);
   context.write(key, result);
  }
```

a. Describe, in natural language, what these lines do; what does this step receive as input; how does the program iterate all values of that key; which is the exact operation performed with these values?

   These lines define the reduce function in the IntSumReducer class. This function receives a key (a word) and a list of values (the counts of that word) as input. It iterates over the list of counts, summing them to get the total count for that word. The sum is then set in the result, which is an IntWritable object. Finally, the function writes the word and its total count to the context, which will output the final result of the word frequency count.

5. Can you find the place where the Sort phase of MapReduce is called/executed? Why not?

   The Sort phase in the MapReduce process is not explicitly called or executed in the WordCount.java code provided. In the Hadoop MapReduce framework, sorting is an internal process that happens automatically between the Map and Reduce phases. It sorts the keys (words in this case) so that all occurrences of the same key are grouped together before being passed to the Reduce phase. This is managed by the framework itself and does not need to be implemented by the developer, hence there are no lines of code in the WordCount.java program that explicitly call the Sort phase.

Question Set P: MapReduce Execution

1. Where does the input file(s) reside, in the local file system or in the HDFS file system? Why?

   The input files job typically reside in the HDFS file system. This is because HDFS is designed to handle large data sets reliably and to stream those data sets at high bandwidth to user applications.

2. Where is the output file stored, in the local file system or in the HDFS file system? Why?

   The output file should also stored in the HDFS file system. HDFS is used for output to ensure that the data is written in a fault-tolerant manner and is accessible for further processing or analysis within the Hadoop ecosystem.

3. Where does your java program reside, in the local file system or in the HDFS file system? Why?

   The Java program, typically a JAR file, resides on the local file system. It is executed by the Hadoop framework which accesses HDFS to read input and write output. The program itself does not need to be in HDFS because it is not the data being processed but the logic that processes the data.

Question Set Q: Adapting the Program

1. Do you have to adapt the map step or the reduce step?

   For WordCountImproved.java, you need to adapt the map step to handle case insensitivity and remove non-letter characters.
   For InitialLetterCount.java, the map step needs to be adapted to emit the initial letter of each word.
   For WordLength.java, the map step should be adapted to emit the length of each word.

2. Which line(s) do you have to adapt? How?

   WordCountImproved.java:

```
while (itr.hasMoreTokens()) {
        // Convert token to lowercase and remove non-letter characters from beginning and
end
        String token = itr.nextToken().toLowerCase().replaceAll("^\\P{L}+",
"").replaceAll("\\P{L}+$", "");
        if (!token.isEmpty()) {
          word.set(token);
          context.write(word, one);
        }
```

   InitialLetterCount.java:
   a. Change the map step to emit the first character of each word.

```
public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString());
```

```
      while (itr.hasMoreTokens()) {
        String token = itr.nextToken().toLowerCase().replaceAll("^\\P{L}+",
"").replaceAll("\\P{L}+$", "");
        if (!token.isEmpty()) {
          firstLetter.set(token.substring(0, 1));
          context.write(firstLetter, one);
        }
```

WordLength.java:
a. Change the map step to emit the length of each word.

```
public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        String token = itr.nextToken().toLowerCase().replaceAll("^\\P{L}+",
"").replaceAll("\\P{L}+$", "");
        if (!token.isEmpty()) {
          wordLength.set(token.length());
          context.write(wordLength, one);
        }
```

b. This line converts the length of each word to an IntWritable object and emits it for counting.

a. For each new program, list the changed lines of source code.

For WordCountImproved.java

```
public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        // Convert token to lowercase and remove non-letter characters from beginning and end
        String token = itr.nextToken().toLowerCase().replaceAll("^\\P{L}+",
"").replaceAll("\\P{L}+$", "");
        if (!token.isEmpty()) { // Check if token is not empty after trimming non-letter characters
          word.set(token);
          context.write(word, one);
        }
```

For InitialLetterCount.java:

```
public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
```

```
    while (itr.hasMoreTokens()) {
     String token = itr.nextToken().toLowerCase().replaceAll("^\\P{L}+",
"").replaceAll("\\P{L}+$", "");
     if (!token.isEmpty()) {
       firstLetter.set(token.substring(0, 1));
       context.write(firstLetter, one);
     }
    }
```

For WordLength.java:

```
public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
     String token = itr.nextToken().toLowerCase().replaceAll("^\\P{L}+",
"").replaceAll("\\P{L}+$", "");
     if (!token.isEmpty()) {
       wordLength.set(token.length());
       context.write(wordLength, one);
     }
    }
```

c. Describe in natural language what these lines do.

For WordCountImproved.java:

The code takes each token generated by the tokenizer, converts it to lowercase to ensure that the counting of words is case-insensitive (meaning that "Hello" and "hello" are considered the same word). Then it removes any leading or trailing characters that are not letters, which means punctuation and numbers attached to the words at the beginning or end are removed. For instance, "World!" and "World" would be considered the same word and counted together.

For InitialLetterCount.java:

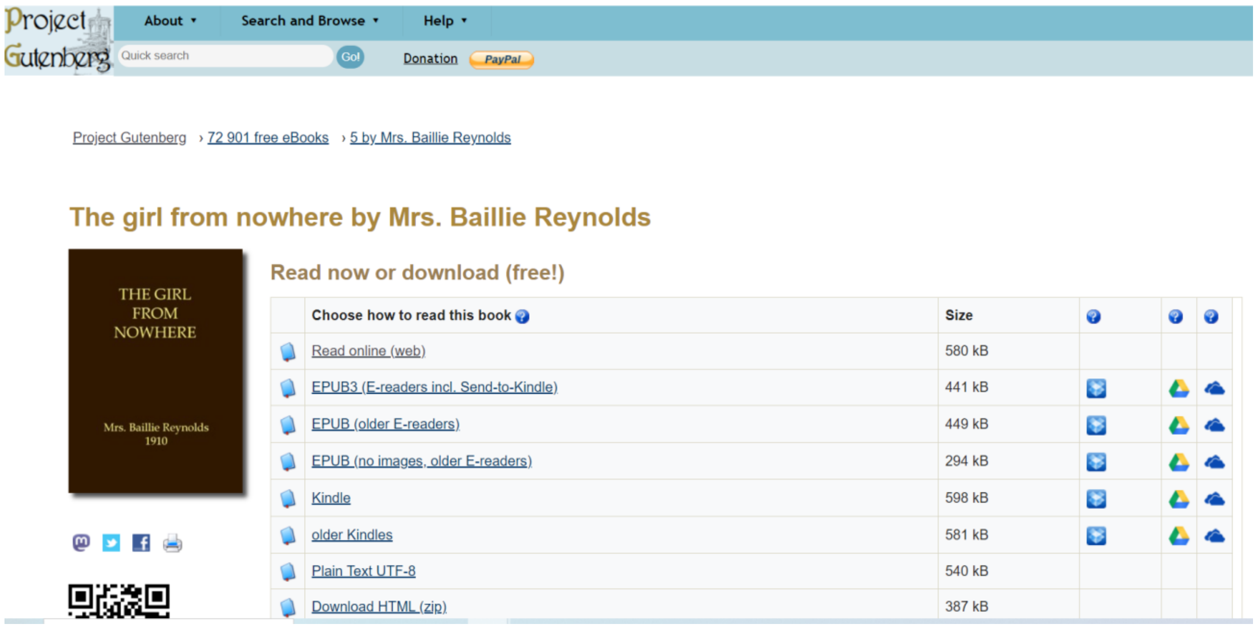
The modified code examines each token and checks if it's not an empty string. If the token is a valid word, it extracts the first character of the word, converts it to uppercase, and writes this initial letter as the key to the context, with a value of 1. This way, the MapReduce framework will count the occurrences of each initial letter across all the input text, regardless of case.

For WordLength.java:

The adjusted code looks at each token and creates a new IntWritable object that represents the length of the token (how many characters it contains). This length is written to the context as the key, with a value of 1. The MapReduce framework then counts how many times words of each length appear in the input text. For example, if the input contains the words "see" and "bee", the number 3 would be output with a count of 2, since both words are 3 characters long.

Question Set R: Large Data Set

1. Describe the input data briefly, e.g., source and size.



Project Gutenberg › 72 901 free eBooks › 5 by Mrs. Baillie Reynolds

The girl from nowhere by Mrs. Baillie Reynolds

Read now or download (free!)

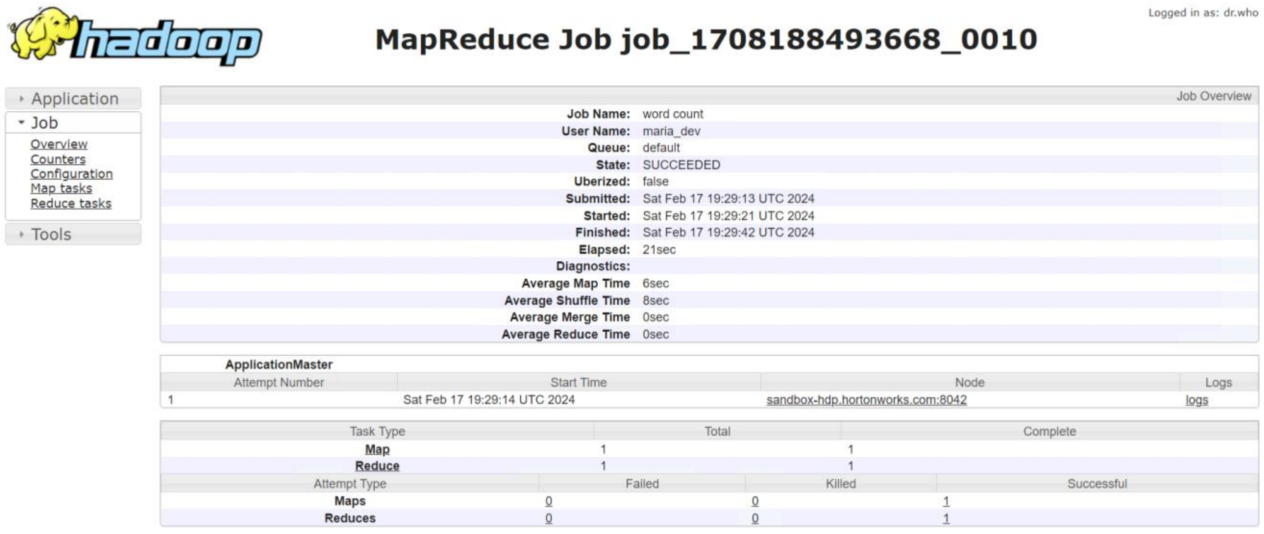
Choose how to read this book	Size
Read online (web)	580 kB
EPUB3 (E-readers incl. Send-to-Kindle)	441 kB
EPUB (older E-readers)	449 kB
EPUB (no images, older E-readers)	294 kB
Kindle	598 kB
older Kindles	581 kB
Plain Text UTF-8	540 kB
Download HTML (zip)	387 kB

The input data for your task is the book "The girl from nowhere" by Mrs. Baillie Reynolds, which is in the public domain and available on Project Gutenberg. The plain text UTF-8 version is approximately 540 kB in size.

For each program:

1. List a short selection of the output.

For WordCountImproved.java:
WCI1:



Logged in as: dr.who

MapReduce Job job_1708188493668_0010

Application
Job
Overview
Counters
Configuration
Map tasks
Reduce tasks
Tools

Job Overview

Job Name:	word count
User Name:	maria_dev
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Sat Feb 17 19:29:13 UTC 2024
Started:	Sat Feb 17 19:29:21 UTC 2024
Finished:	Sat Feb 17 19:29:42 UTC 2024
Elapsed:	21sec
Diagnostics:	
Average Map Time	6sec
Average Shuffle Time	8sec
Average Merge Time	0sec
Average Reduce Time	0sec

ApplicationMaster

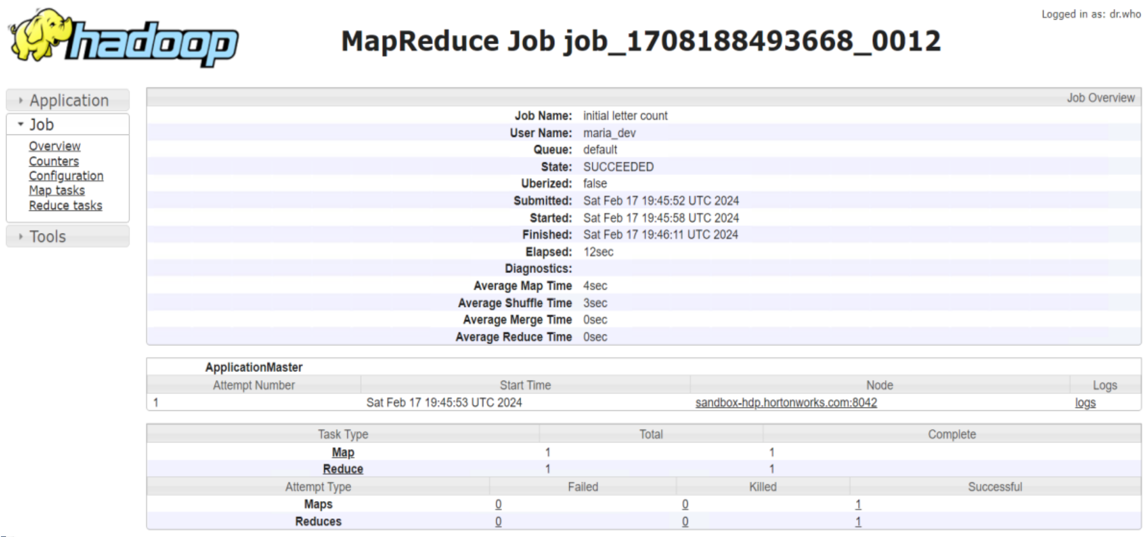
Attempt Number	Start Time	Node	Logs
1	Sat Feb 17 19:29:14 UTC 2024	sandbox-hdp.hortonworks.com:8042	logs

Task Type	Total	Complete
Map	1	1
Reduce	1	1

Attempt Type	Failed	Killed	Successful
Maps	0	0	1
Reduces	0	0	1

```
you?"	17
young	167
young,	12
younger	7
younger,	1
younger.	1
your	180
yours	1
yours,	1
yours--might	1
yours.	1
yours."	1
yourself	10
yourself,	2
yourself,"	1
yourself----"	1
yourself.	3
yourself."	2
yourself?	1
yourself?"	1
yourtar	4
yourtar,	2
yourtars	1
youth	9
youth,	4
youth.	5
zealous,	1
zephyr	1
zest	1
```

For InitialLetterCount.java:
ILC:



MapReduce Job job_1708188493668_0012

Job Overview

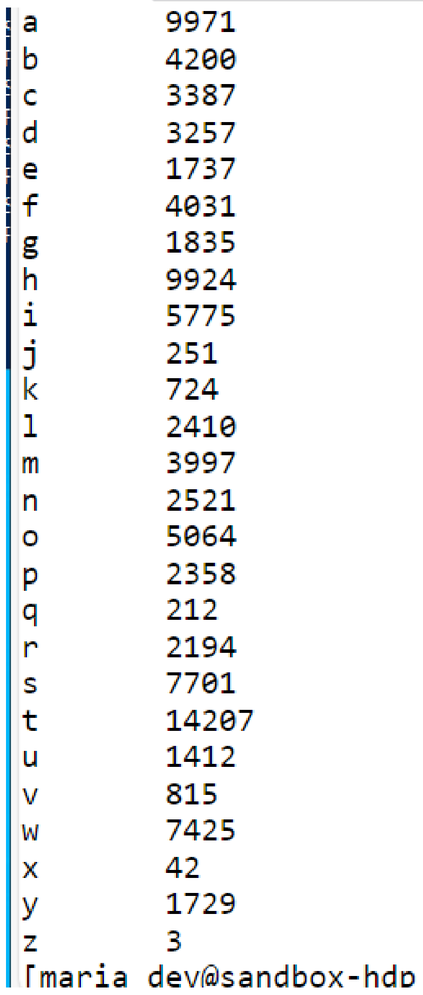
Job Name:	initial letter count
User Name:	maria_dev
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Sat Feb 17 19:45:52 UTC 2024
Started:	Sat Feb 17 19:45:58 UTC 2024
Finished:	Sat Feb 17 19:46:11 UTC 2024
Elapsed:	12sec
Diagnostics:	
Average Map Time	4sec
Average Shuffle Time	3sec
Average Merge Time	0sec
Average Reduce Time	0sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Sat Feb 17 19:45:53 UTC 2024	sandbox-hdp.hortonworks.com:8042	logs

Task Type	Total	Complete
Map	1	1
Reduce	1	1

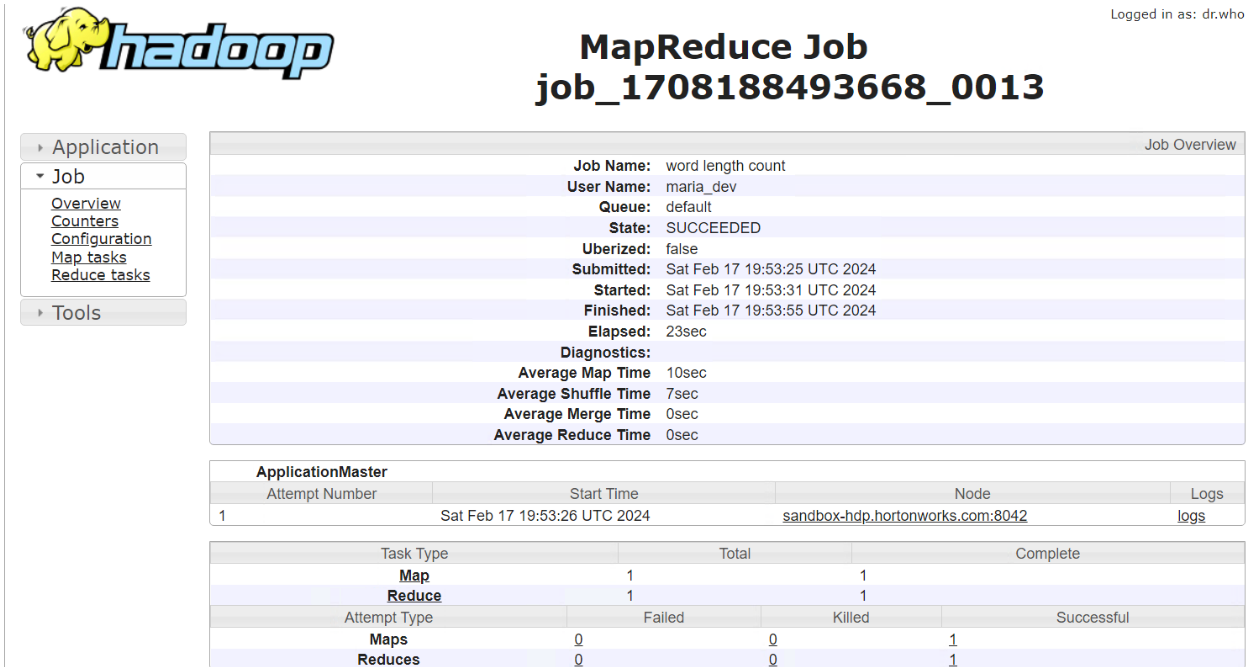
Attempt Type	Failed	Killed	Successful
Maps	0	0	1
Reduces	0	0	1



```
a       9971
b       4200
c       3387
d       3257
e       1737
f       4031
g       1835
h       9924
i       5775
j       251
k       724
l       2410
m       3997
n       2521
o       5064
p       2358
q       212
r       2194
s       7701
t       14207
u       1412
v       815
w       7425
x       42
y       1729
z       3
[maria_dev@sandbox-hdp
```

For WordLength.java:

WL:



MapReduce Job job_1708188493668_0013

Job Overview

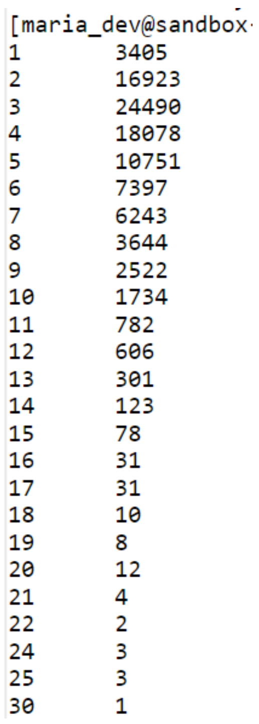
Job Name:	word length count
User Name:	maria_dev
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Sat Feb 17 19:53:25 UTC 2024
Started:	Sat Feb 17 19:53:31 UTC 2024
Finished:	Sat Feb 17 19:53:55 UTC 2024
Elapsed:	23sec
Diagnostics:	
Average Map Time	10sec
Average Shuffle Time	7sec
Average Merge Time	0sec
Average Reduce Time	0sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Sat Feb 17 19:53:26 UTC 2024	sandbox-hdp.hortonworks.com:8042	logs

Task Type	Total	Complete
Map	1	1
Reduce	1	1

Attempt Type	Failed	Killed	Successful
Maps	0	0	1
Reduces	0	0	1



```
[maria_dev@sandbox
1       3405
2       16923
3       24490
4       18078
5       10751
6       7397
7       6243
8       3644
9       2522
10      1734
11      782
12      606
13      301
14      123
15      78
16      31
17      31
18      10
19      8
20      12
21      4
22      2
24      3
25      3
30      1
```

2. Describe how/if the execution time and the output file size varied across the programs, and between small and large data.

The execution time and output file size for MapReduce jobs can vary based on several factors, including the complexity of the job, the nature of the data being processed, and the computational resources available. And we can analyze the following:

Execution Time:

The execution time of a MapReduce job can be influenced by the complexity of the map and reduce functions, the amount of data processed, and the efficiency of the data processing (like sorting and shuffling).
A job that processes more complex data or has a more complex logic in the map or reduce functions may take longer. Also, if the data is large, the execution time will generally be longer.

Output File Size:

The output file size is determined by the amount of data emitted by the reduce function. For instance, counting initial letters may result in a smaller output file compared to counting word occurrences since there are fewer unique initial letters than there are unique words or unique word lengths.
The granularity of the output (such as individual words vs. word lengths) can also affect the output size, with more granular outputs typically being larger.