Hands-On: Installing Cloudera Manager Server, Health Check, Monitoring, and Testing

This comprehensive guide covers the installation of Cloudera Manager Server, followed by health checks, monitoring, and testing in a production environment.

Installation of Cloudera Manager Server

Prerequisites:

- Supported Linux distribution (CentOS, Red Hat Enterprise Linux, etc.).
- Sufficient hardware resources (CPU, RAM, Disk space).
- Root or sudo privileges.
- Network configured with a static IP and proper DNS settings.
- Java Development Kit (JDK) installed.

Step 1: System Preparation

- Update your system packages.

```
sudo yum update -y
```

- Install Java Development Kit (JDK) if not already installed.

```
sudo yum install java-1.8.0-openjdk-devel
```

Step 2: Configure Cloudera Repository

- Add the Cloudera repository.

```
wget [Cloudera Repository URL] -O /etc/yum.repos.d/cloudera-manager.repo
```

Step 3: Install Cloudera Manager Server

- Install Cloudera Manager Server and daemons.

```
sudo yum install cloudera-manager-daemons cloudera-manager-server
```

Step 4: Database Setup

- Opt for an external database like PostgreSQL, MySQL, or Oracle for production use.

- Configure the database for Cloudera Manager.

```
sudo /opt/cloudera/cm/schema/scm_prepare_database.sh [db_type] [db_name] [db_username]
```

Step 5: Start the Server

- Start the Cloudera Manager Server.

```
sudo systemctl start cloudera-scm-server
```

Step 6: Access Cloudera Manager

- Access the Cloudera Manager web interface at `http://[your-server-ip]:7180`.
- Complete the installation wizard, selecting services and configurations as per your requirements.

Health Check and Monitoring

Post-Installation Health Check

- Check the status of Cloudera Manager Server.

  ```
  sudo systemctl status cloudera-scm-server
  ```

- Validate the log files for any errors or warnings.

  ```
  cat /var/log/cloudera-scm-server/cloudera-scm-server.log
  ```

Monitoring Cloudera Manager

- **Resource Utilization:** Regularly monitor CPU, memory, and disk usage.
- **Network Connectivity:** Ensure all nodes in the cluster are communicating correctly.
- **Service Health:** Use Cloudera Manager's dashboard to monitor the health of all services.
- **Alerts and Notifications:** Configure alerts for any critical events or thresholds.

Testing

1. Load Testing

Load testing involves simulating real-world load scenarios to assess how the system performs under heavy usage.

a. Hadoop MapReduce Job for Load Testing: - Objective: Run a MapReduce job to stress-test Hadoop's processing capabilities. **- Script Example:** `bash hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar pi 50 1000` This command runs a sample MapReduce job that calculates the value of Pi with 50 maps and 1000 samples.

b. Spark Load Test: - Objective: Test the performance of Apache Spark under heavy data processing load. **- Script Example:** `scala val testRdd = sc.parallelize(1 to 10000000) testRdd.map(x => (x, x)).reduceByKey(_ + _).collect()` This Spark script generates a large RDD and performs a reduce operation.

2. Performance Benchmarking

Benchmarking involves running specific workloads to measure the performance of various components in your CDP setup.

a. TPC Benchmarks: - Objective: Use TPC benchmarks to measure the performance of your SQL-on-Hadoop solutions like Hive or Impala. - **Action:** Run TPC-DS or TPC-H benchmarks using Hive or Impala to test query performance. `sql SELECT COUNT(*) FROM store_sales, store_returns WHERE ss_ticket_number = sr_ticket_number;` This example query is a part of the TPC-DS benchmark suite.

3. Failover and Recovery Testing

Test the resilience of your system by simulating failures and practicing recovery procedures.

a. HDFS Failover Test: - Objective: Simulate a NameNode failure to test failover to a standby NameNode. - **Action:** `bash sudo -u hdfs hdfs haadmin -failover nn1_hostname nn2_hostname` This command forces a failover from the active NameNode (`nn1_hostname`) to the standby NameNode (`nn2_hostname`).

b. Disaster Recovery Drill: - Objective: Test the disaster recovery plan by simulating a data center outage. - **Action:** - Backup critical data using tools like `distcp` or snapshots. - Restore the data in a different cluster/environment and validate.

4. Security Testing

Perform rigorous security tests to identify and address potential vulnerabilities.

a. Kerberos Authentication Test: - Objective: Verify that Kerberos authentication is working correctly across the cluster. - **Script Example:** `bash kinit -kt /path/to/keytab_file principal_name hadoop fs -ls /` This script obtains a Kerberos ticket and then performs a basic Hadoop filesystem operation to check authentication.

b. Vulnerability Scanning: - Objective: Scan for vulnerabilities in the network and applications. - **Tool Suggestion:** Use tools like Nessus or OpenVAS for network vulnerability scanning.

c. Penetration Testing: - Objective: Conduct penetration testing to identify security weaknesses. - **Action:** Engage a professional security team to perform controlled penetration tests against your environment.

Best Practices

1. **Automate Testing:** Wherever possible, automate testing procedures for efficiency and consistency.

2. **Continuous Monitoring:** During and after tests, continuously monitor system metrics and logs.
3. **Incident Reporting:** Document any incidents or anomalies encountered during testing.
4. **Regular Audits:** Conduct regular audits of your system post-testing to ensure ongoing compliance and performance.
5. **Stress Testing:** Occasionally perform stress tests that go beyond normal operational capacity to understand system limits.