
# Deep dive into the world of federated searches

Raanan Dagan, Principal SE Architect, Splunk
Sourav Pal, Senior Principal Engineer, Splunk

Oct 2018

# Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.



splunk> .conf18

Data Fabric Search is a new, extended search platform that leverages compute assets from anywhere and accesses and executes on data regardless of origin or type.

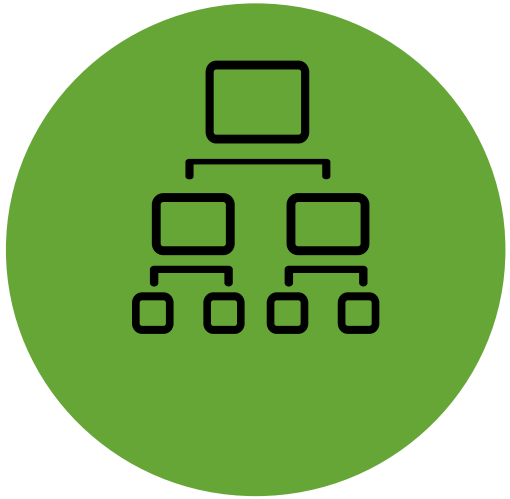
splunk> .conf18

# Features

# Two main features of DFS

## Big Data Analysis

*Processing massive amounts of data*



## Federated Searches

*Reaching data where it resides*

- Other Splunk Deployments

# Big Data Analysis

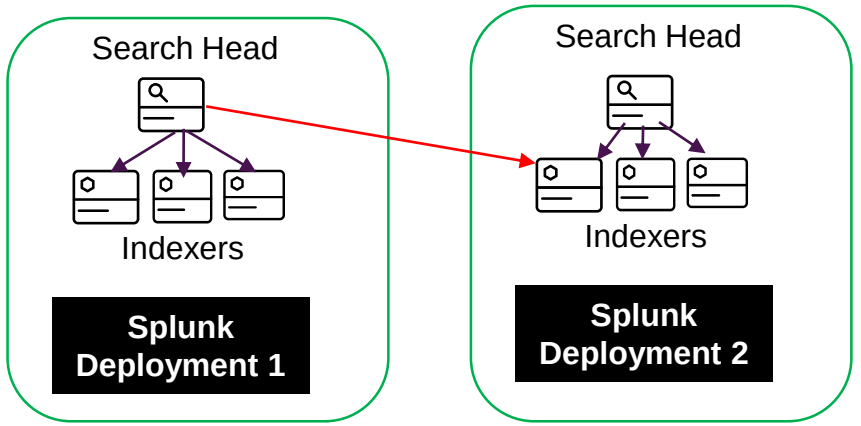
*Processing massive amounts of data*

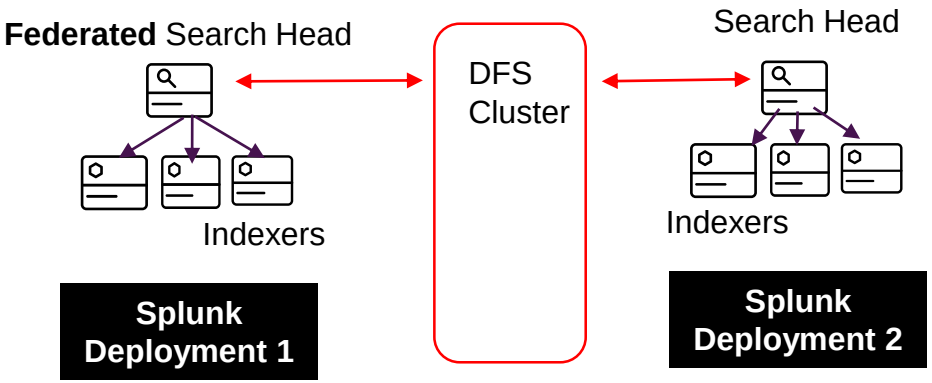
- **Current Scenario**:
  - Ex:- 40 mil cardholders
    - Stats (high cardinality); join / union (data mashup); sorting
  - Scale Constraints
    - When using Splunkd, the join commands is limited to 50,000 events in sub-search
- **Solution:**
  - DFS has a powerful query orchestration engine that:
    - Leverages Splunkd for first mile compute
    - Workload offloaded from Splunk Search Head enabling better performance of the system
    - Leverages DFS SearchPipeline for large-scale parallel and distributed data processing on compute cluster (currently Spark)
    - Ex:  handled searches successfully which have resulted in output of  267 billion events.

# Federated Searches across Splunk

*Reaching data where it resides - other Splunk Deployments*

- **Current scenario with multiple deployments**:
  - Unified Search across Splunk Deployments
    - Requires search head directly interact with Indexers of other deployments
    - Bypassing security configuration and resource usage of other deployments
    - Need replication of knowledge objects to to provide a unified view across all search heads
- **Federated Search**:
  - Seamlessly search across all deployments
    - Ability to correlate and run join/unions to search across datasets in disparate deployments
    - Enables scale and improves the performance by offloading the workload from search head to Spark worker nodes
    - Better management of security configurations



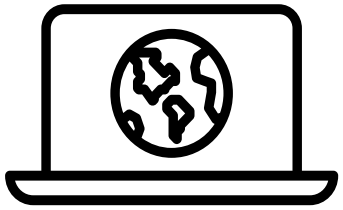



# When do I use DFS?

## **Customer ABC** has three LoBs

**Video**

- 20 mil subscribers



**Internet**

- 18 mil subscribers

**Wireless**

- 11 mil subscribers

- Each LoB has a separate Splunk deployment
- Outage over the weekend affecting customers in California

# DFS Use-Cases

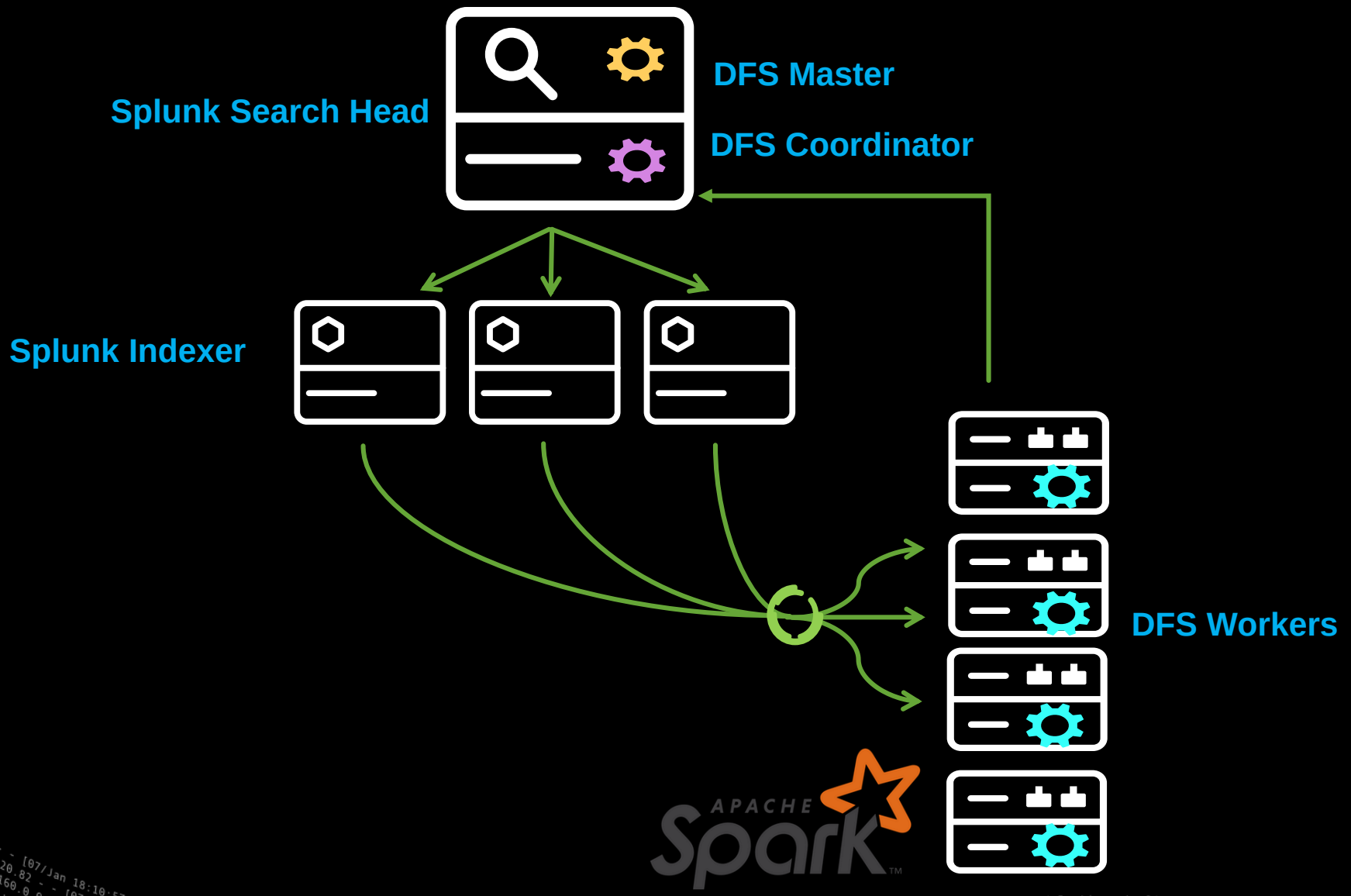
- Use-Case 1: Massive Scale Computation
  - Number of “Video” Customers that were affected by this outage.
  - Index = Video | **stats** count (customer_id) OR ip_address where state is CA

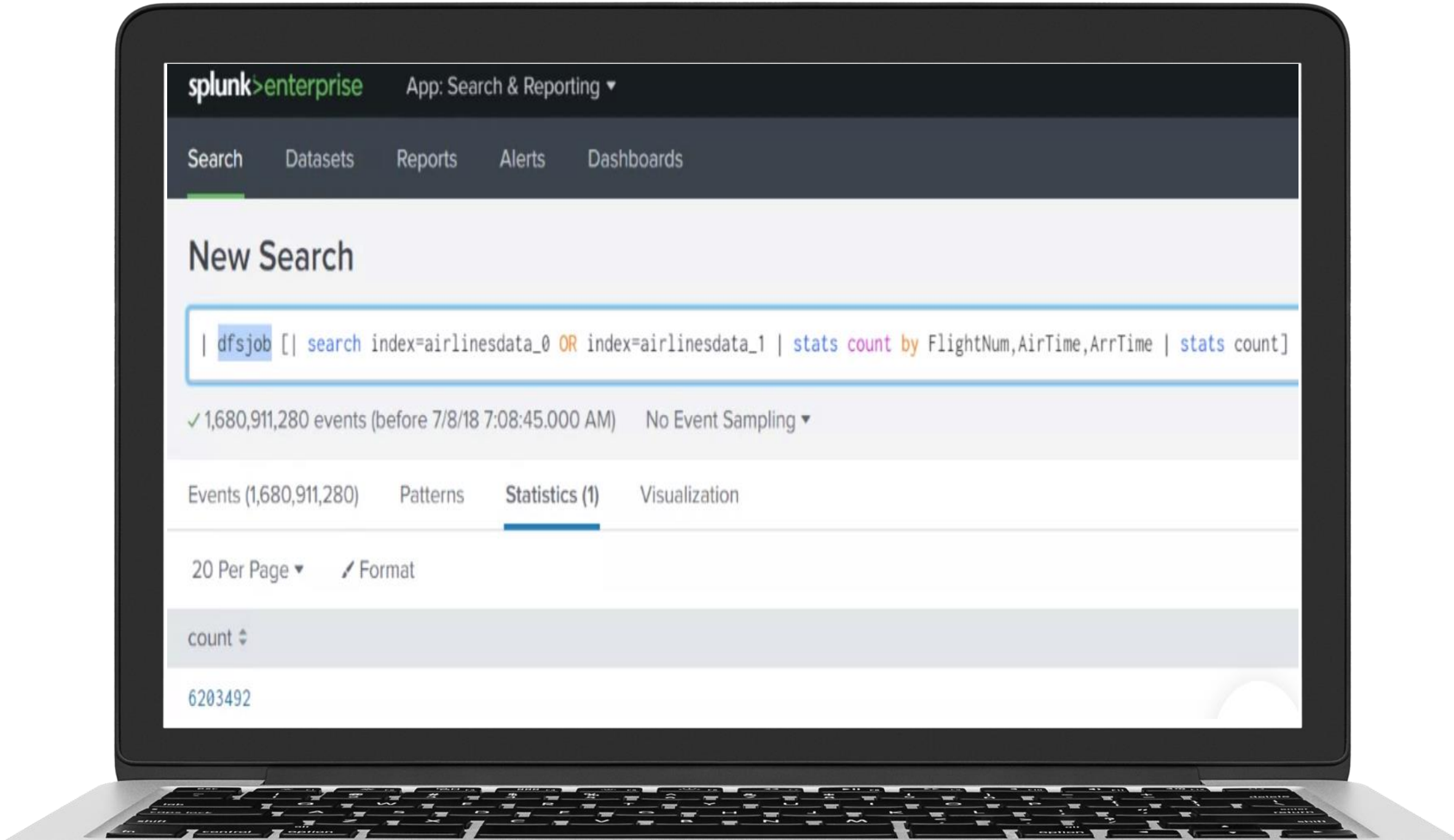
- Use-Case 2: Federated Searches (Splunk Deployments)
  - Number of Video AND Internet customers that were affected by this outage
  - Index = Video | **join index=Internet**

# Technical Deep Dive

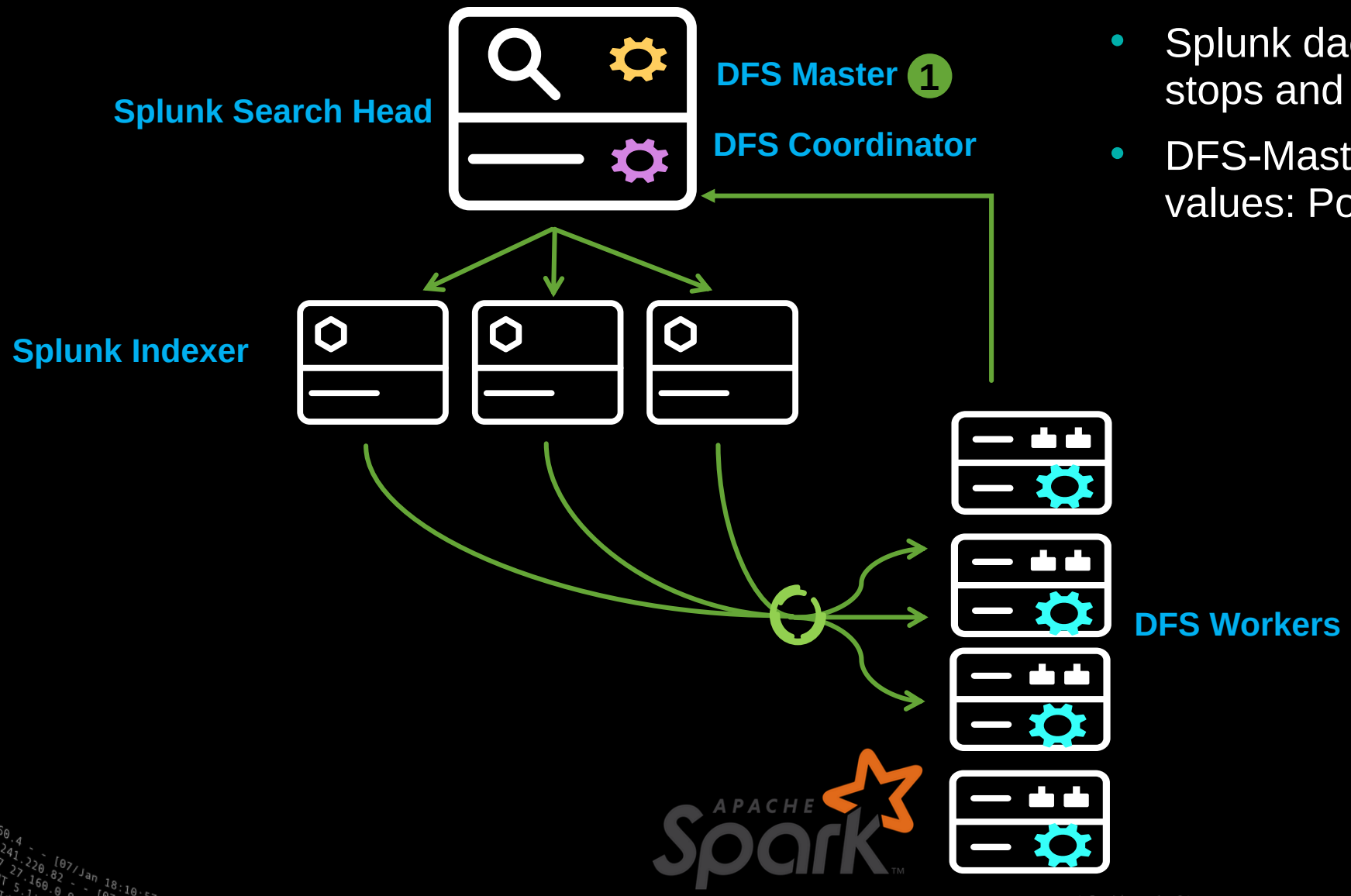
splunk> .conf18

# Big Data Analysis



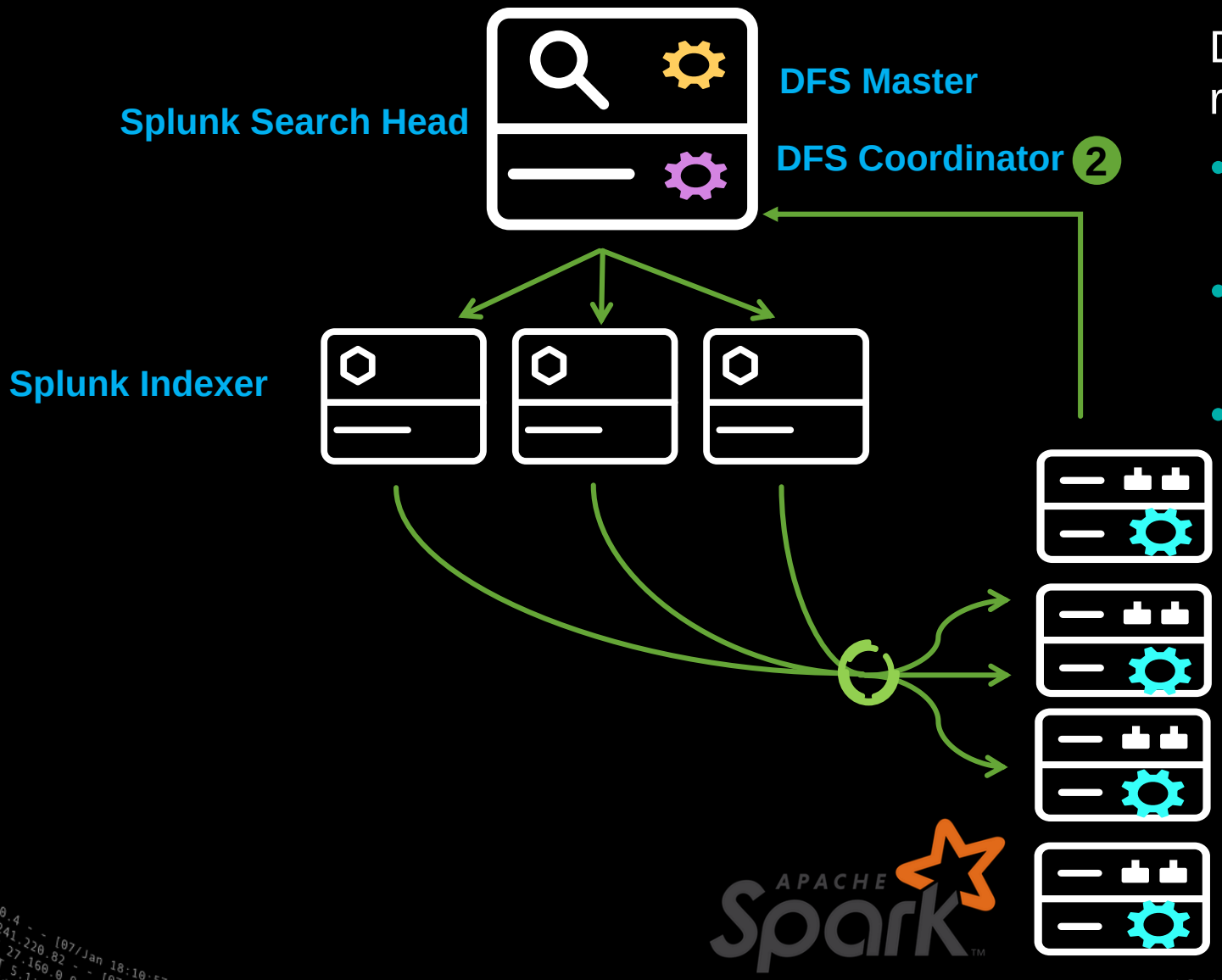


# Step 1: Life Cycle Management of Data Fabric Search Master




- Splunk daemon process stop / start also stops and starts DFS-Master
- DFS-Master is initialized with several values: Ports, Number of Executors, etc

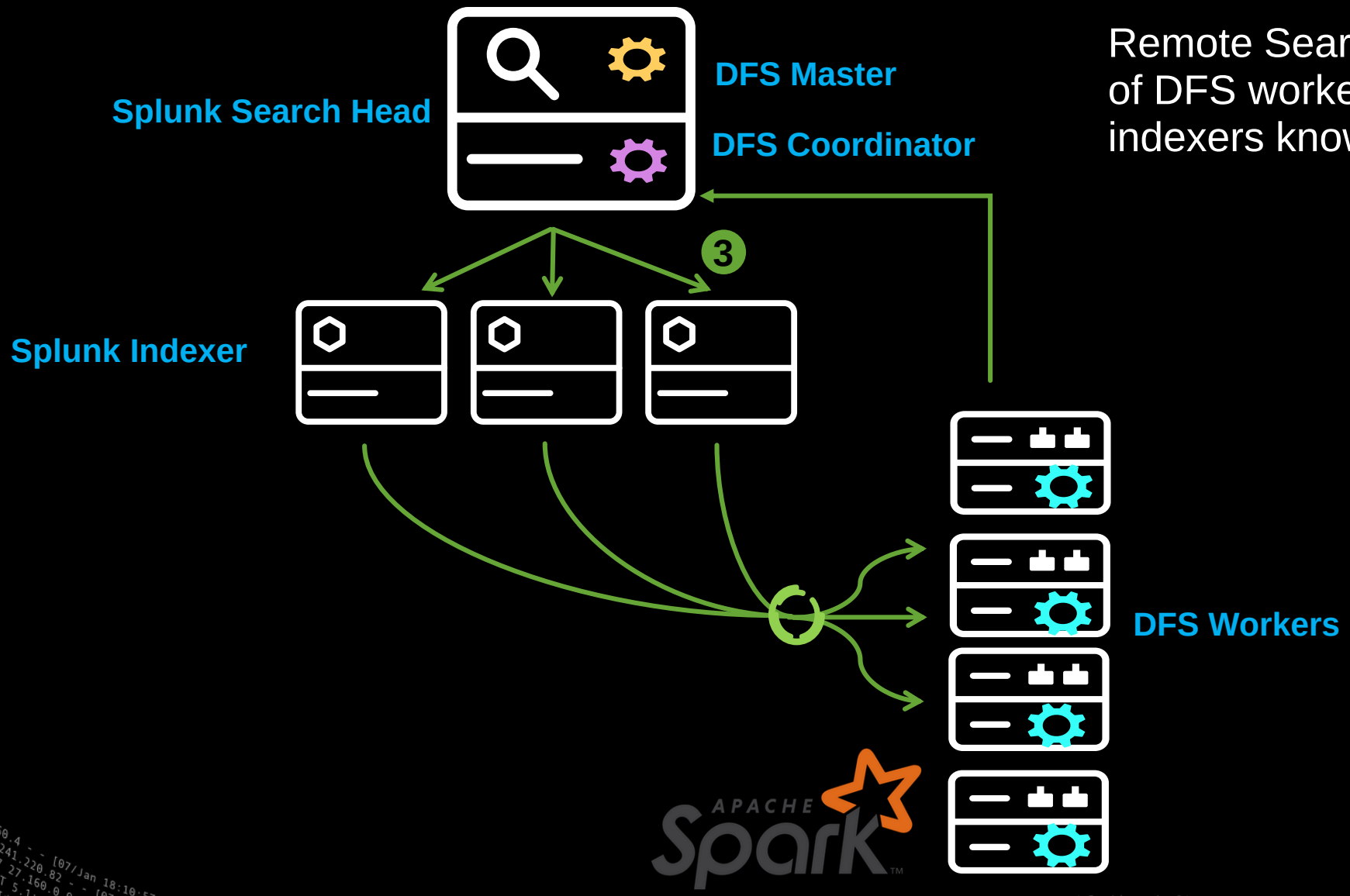
# Step 2: DFS Query Planning




DFS Coordinator partition the search requested into 3 phases:

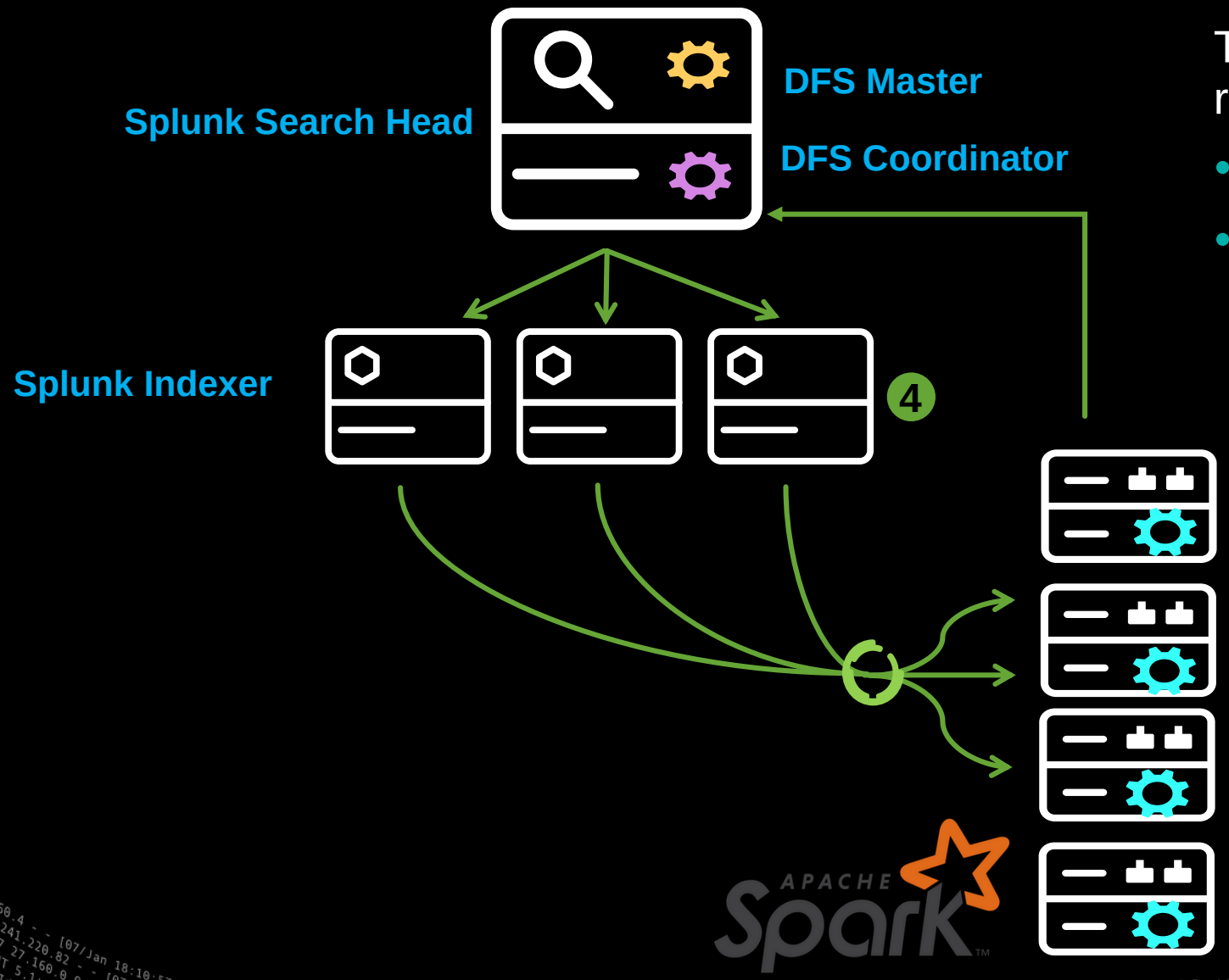
- Splunk Indexer Job aka Remote Search Job
- DFS job or the execution which will be executed on the spark cluster
- Search Head (SH) job - which will be executed if any on the SH

# Step 3: Dynamic Partition Setup and Query Update




Remote Search job is updated with the list of DFS workers listeners such that the indexers know where to send the data

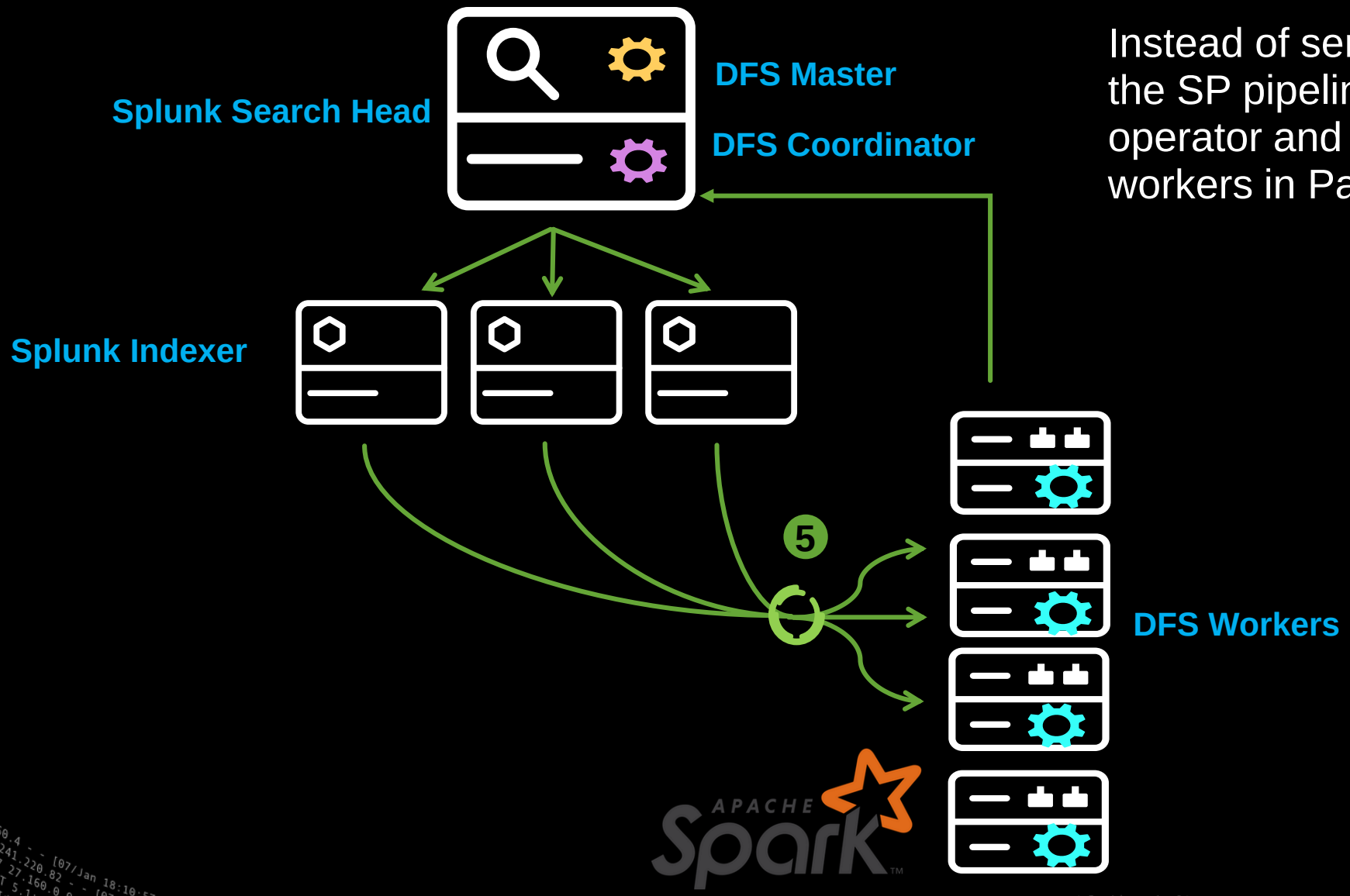
# Step 4: Splunk Indexer layer execution




The remote search processes execute the remote pipeline

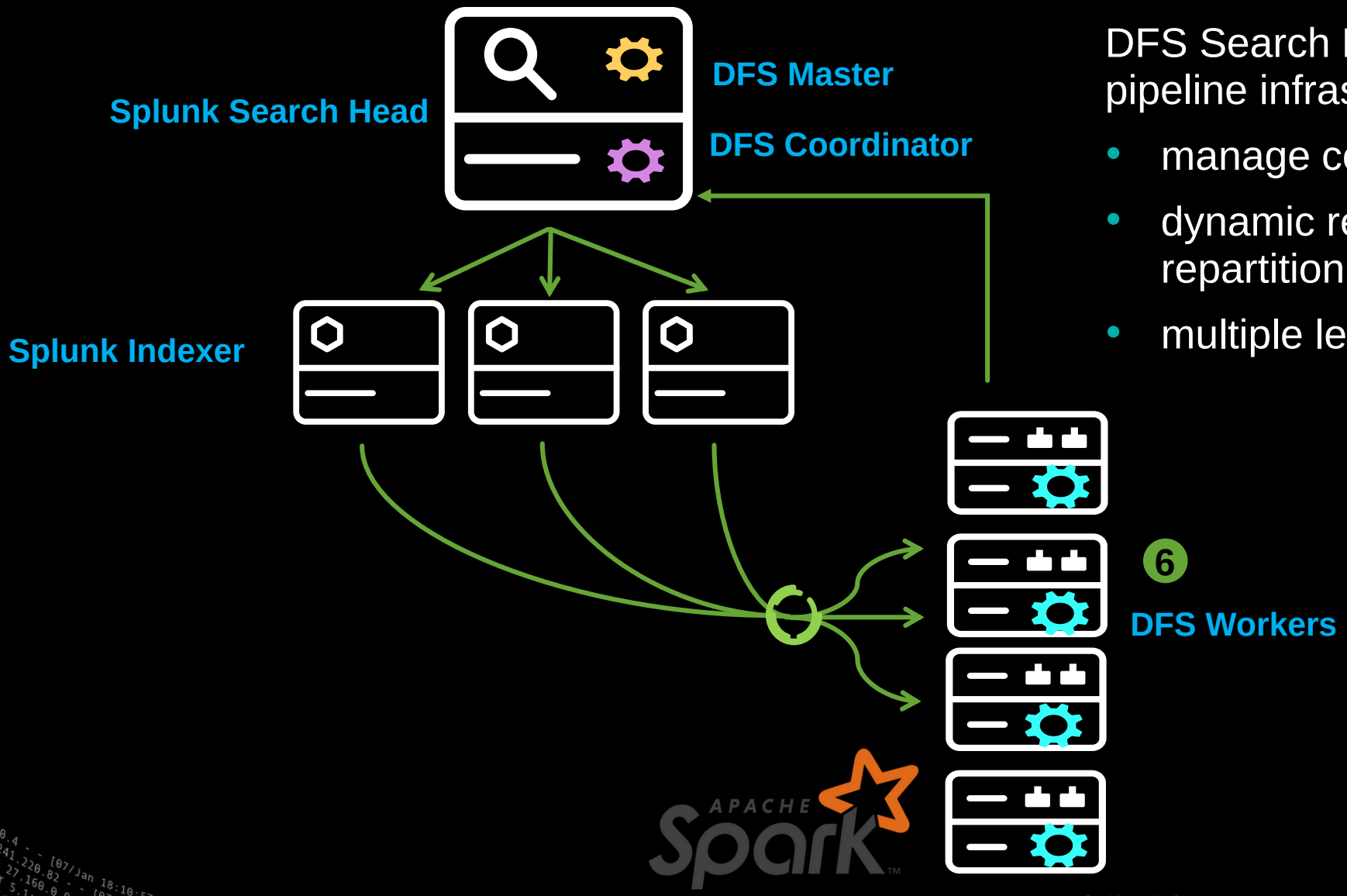
- reading data from the buckets
- applying extractions, kv, lookups, and other streaming search processors (pre-stats)

# Step 5: Shuffle to DFS workers




Instead of sending the data back to the SH, the SP pipeline executed redistribute operator and shuffle the data to the DFS workers in Parallel (RD_OUT)

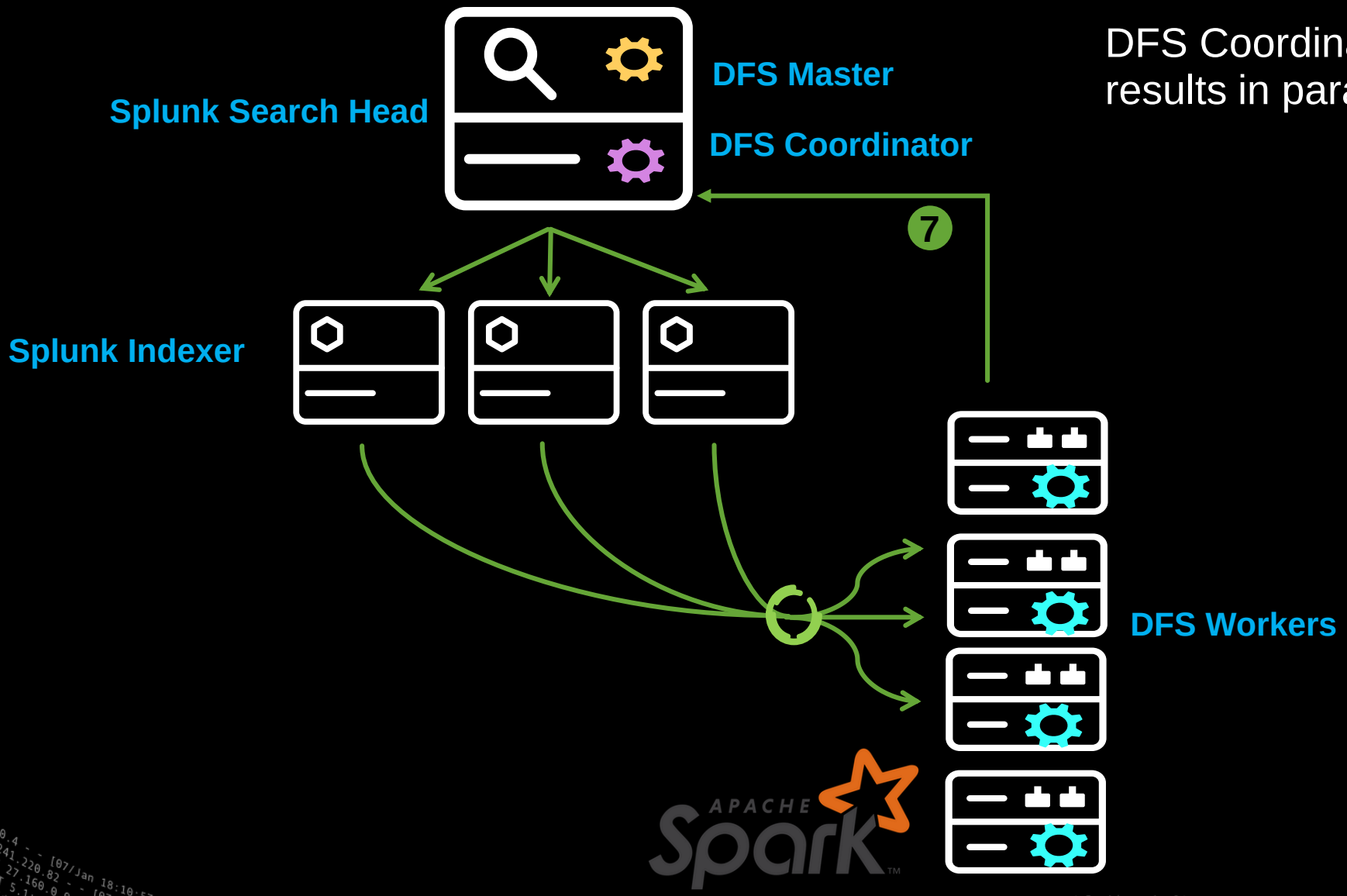
# Step 6: DFS Workers Job execution




DFS Search Pipeline are Java lambdas and pipeline infrastructures

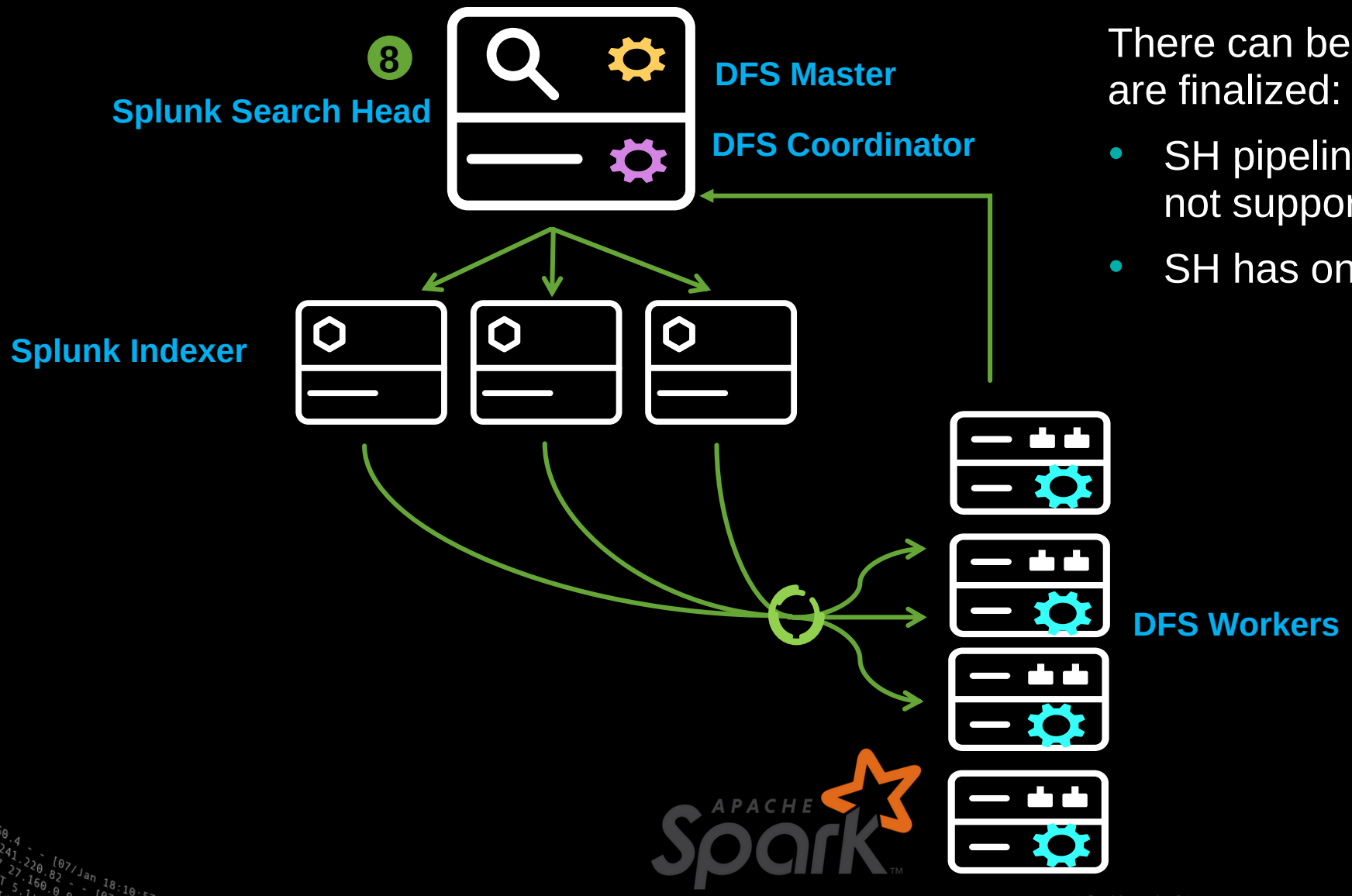
- manage core scaling (memory or disk)
- dynamic repartitioning (Spark RDD repartitioning) based on data skews
- multiple levels of shuffle as needed

# Step 7: Parallel Data Retrieval




DFS Coordinator retrieves the search results in parallel from the DFS workers

# Step 8: SH Job execution and Job finalization




There can be two scenarios before results are finalized:

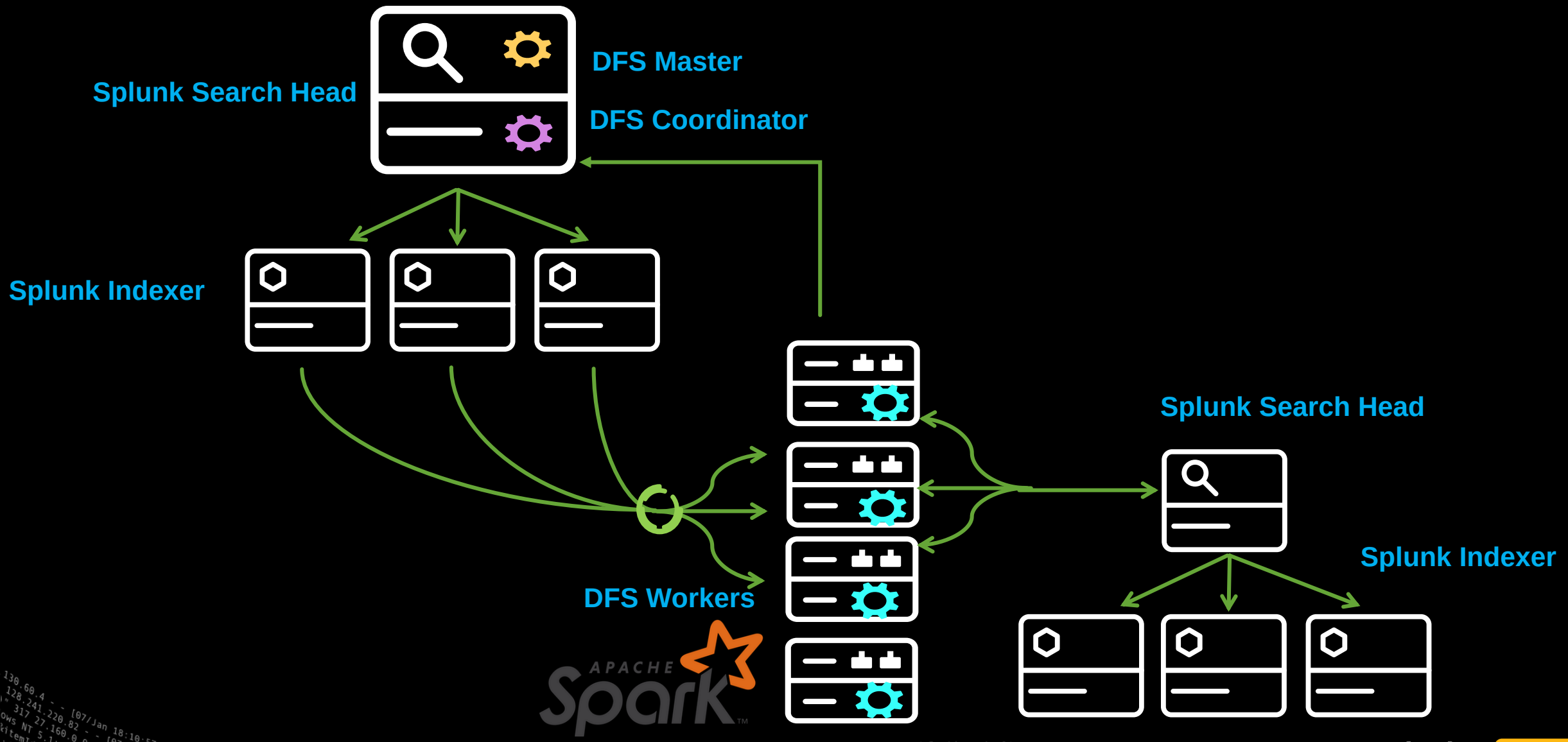
- SH pipeline has processors which were not supported in DFS pipeline
- SH has only the collection processor

# DFS – Performance

**JOIN and Stats 1.5 trillion record**

- **1,500,000,000,000 in-flight, not at-rest**
- **Ran in ~50 minutes on a ~100 node Spark cluster**

# Federated Searches – Splunk Deployments

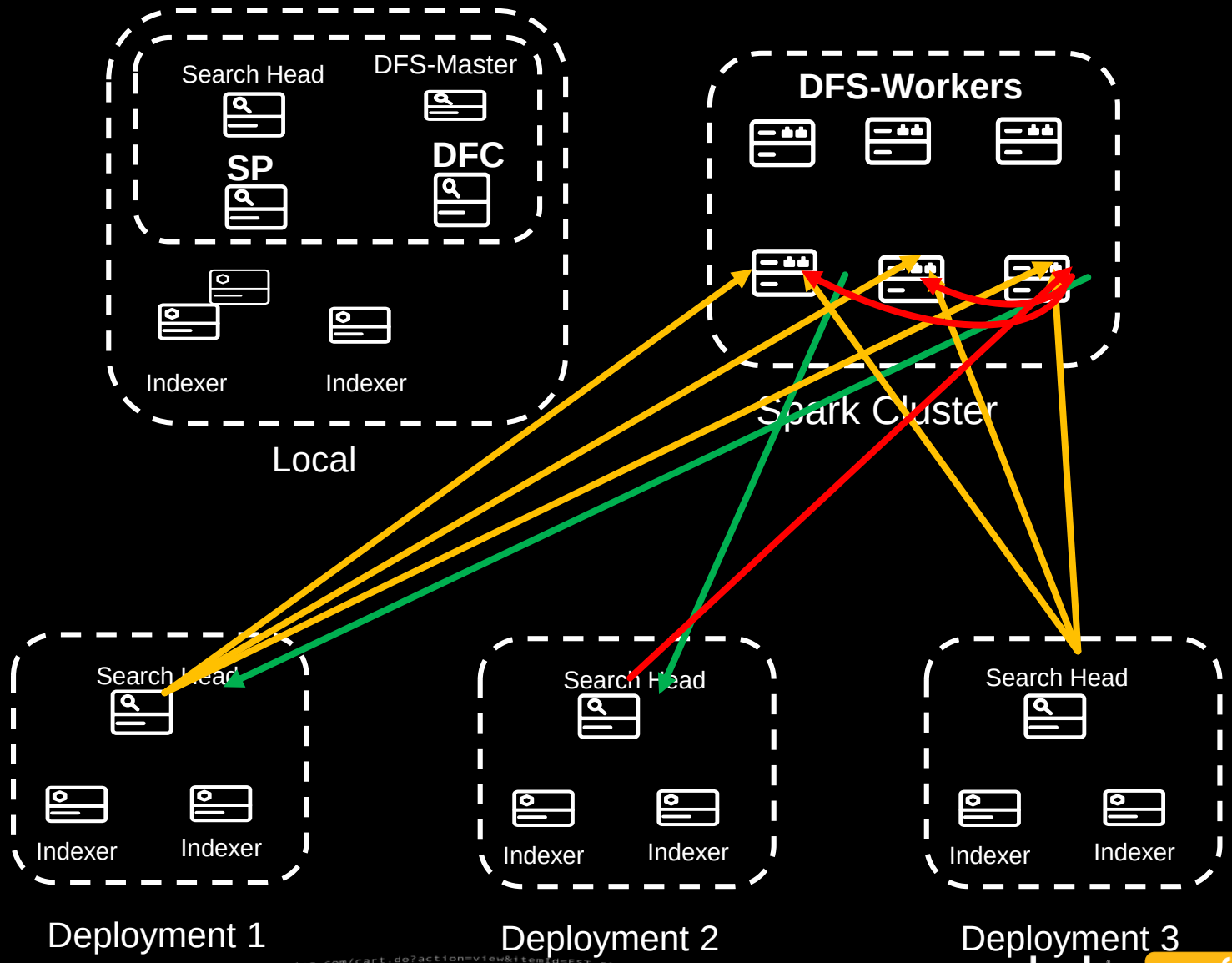


# Federated Search – Execution Flow

**SP**:
Start Query Execution Flow

**DFC**:
Information Map Generation
Access Control
Job Partitioning

**DFS-Workers**: Create Data Layer and Execute Triggers
**Version fetch**
Triggering the query
Sid Management

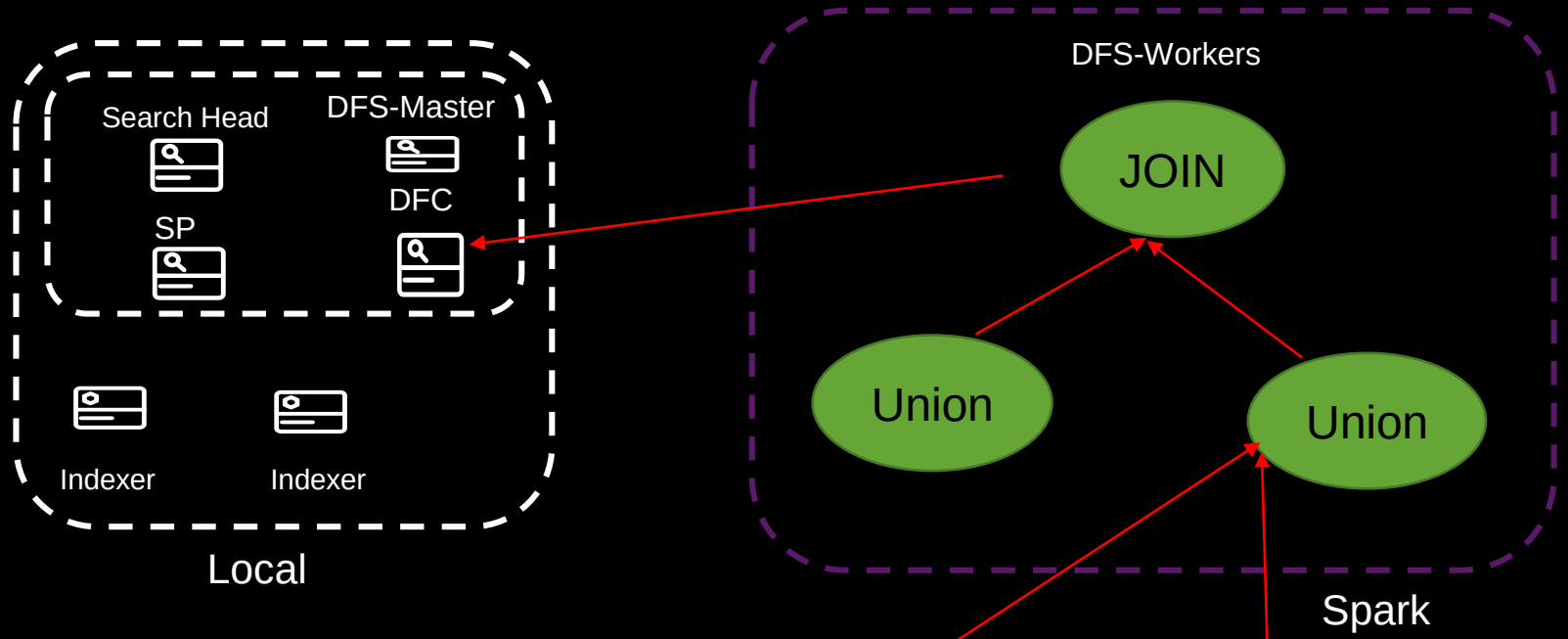


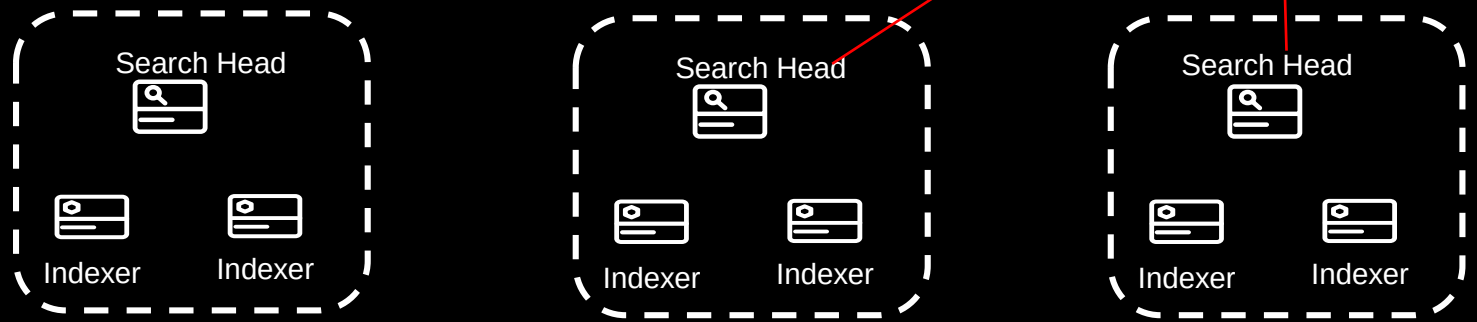

**DFS SH (RD_OUT)**
**Non-DFS SH (REST API)**

# Federated Search – Query Example

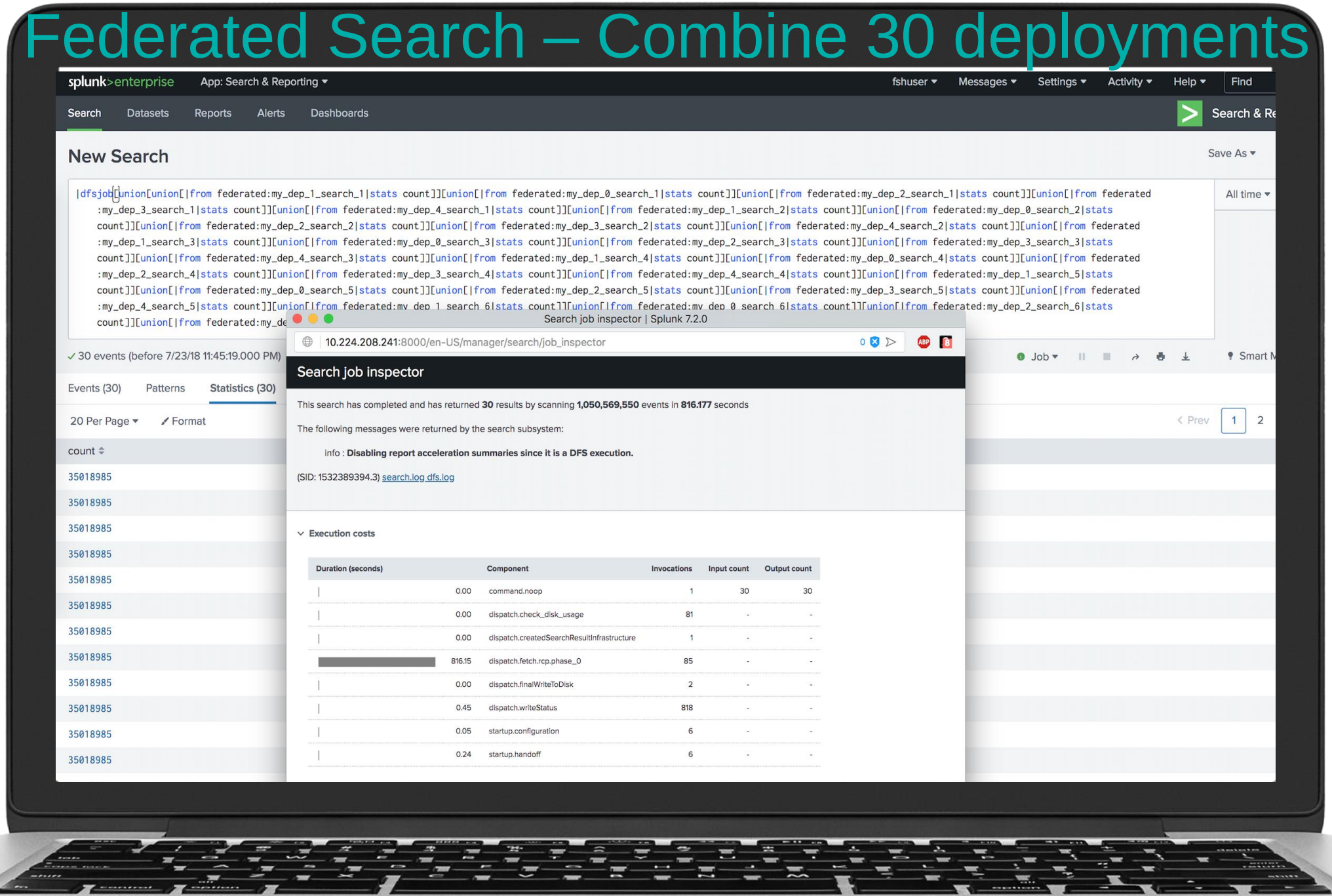
## Join across multiple deployments

```
|dfsjob[
| union [|search Local| ][
|from federated:dep_1]
| join
[| union [|from
federated:dep_2][|from
federated:dep_3]]
|stats count
```



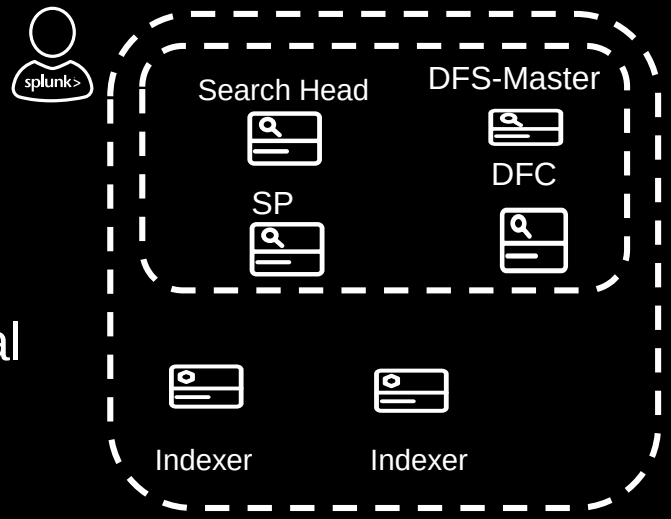
# Federated Search – Combine 30 deployments

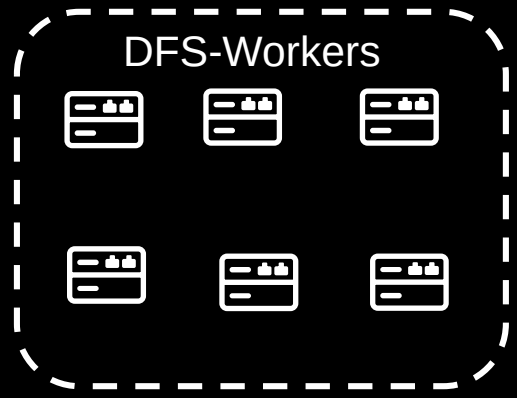


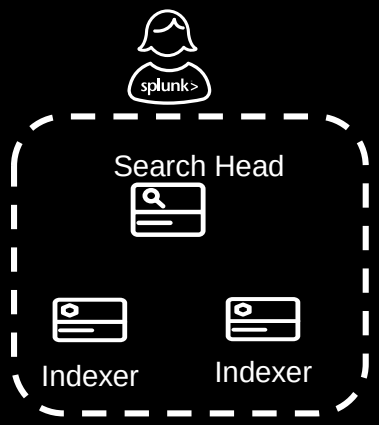
# Federated Search – Access Control Security

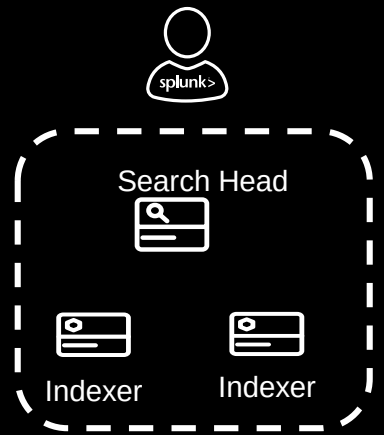
**Service Account Setup**

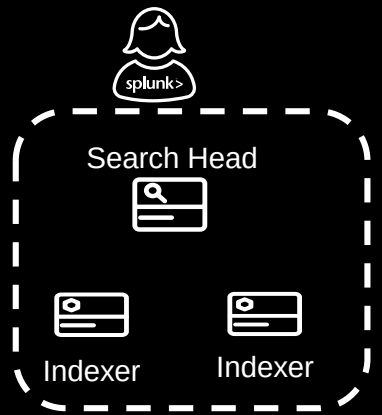
- Remote User and Password = Local User and Password
- Remote Role != Local Role




splunk> .conf18

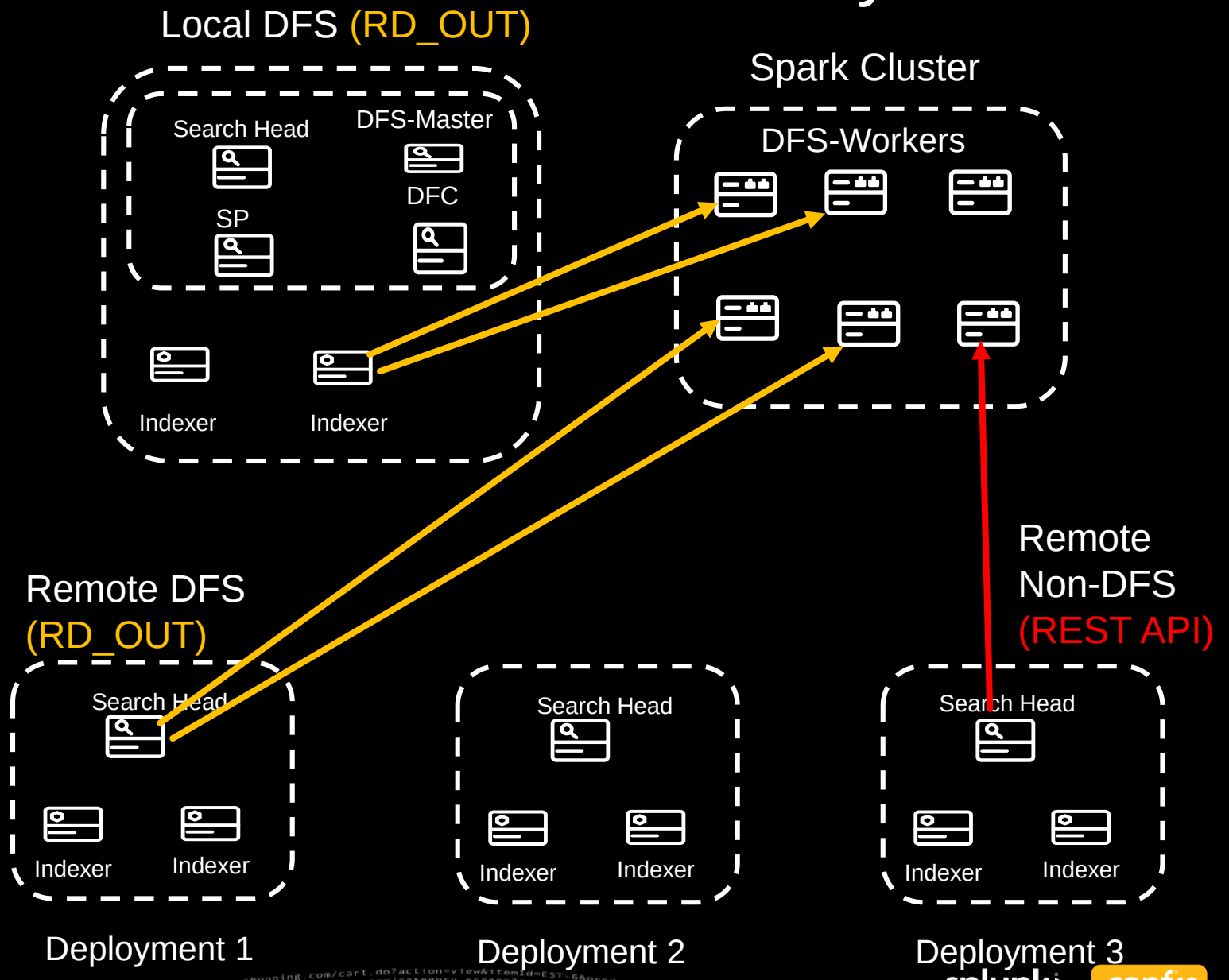
# Federated Search (FS) – Security

| Components | Responsibilities |
|---|---|
| **Management via Service Account** | • On Remote Search Heads, create a new service accounts (user/password) and roles to limit the set of indexers and other resources<br>• This new remote service account will need to match the DFS Master Search Head service account<br>• Example = Users “remotefsh” and Role “role_myfsh1”<br>• Management via Single Identity Provider (LDAP) is not part of the first release and will be added at a later date |
| **FS feature introduces new roles** | • fsh_manage (Admin), fsh_search (User), fsh_remote_deployment (User)<br>➢ [role_myfsh1]<br>➢ importRoles = user;power<br>➢ fsh_search = enabled<br>➢ fsh_remote_deployment = remote_deployment_1 |
| **FS feature new Password Management** | • encrypted in the fshpassword file<br>• Allows Search, we check if user has either the fsh_search or fsh_manage capabilities<br>• Allows Manage, we check if admins has fsh_manage |
| **Runtime Authorization and Authentication** | • FS retrieves the password for remote deployment and proceeds for execution<br>• The authentication and authorization for the remote deployment is based on the default policies executed on the remote Search Head deployment |

# Federated Search – Network Security

- **Default**: All Network is Secured just like normal Splunk. **No configuration required**.
- Data flows are authenticated and encrypted via default symmetric key over a TLS communication

- **Additional Security**:
- Spark Sasl Encryption,
- Spark key stores certificates,
- DFS key stores in limits.conf/server.conf



# Federated Search – Configuration

| Components | Example |
|---|---|
| **Catalog.conf** | [**remote_deployment_1**]<br>IP = 10.0.0.3<br>ServiceAccount = remotefsh<br>Type = Splunk<br><br>[federated:**my_dep_1_search_1**]<br>search = “search index=myindex \| stats count by ip”<br>deploymentName = **remote_deployment_1** |
| **Authorize.conf** | [role_myfsh1]<br>importRoles = user;power<br>fsh_search = enabled<br>fsh_remote_deployment = **remote_deployment_1** |
| **Search** | \|union [\|from federated:**my_dep_1_search_1**] [\|from federated:my_dep_2_search_2] \| stats count |

# Splunk Demo

# Thank You

**Don't forget to rate this session in the .conf18 mobile app**