Search on Splunk Cloud Platform

Intro to SPLv2, the module system, and the catalog

Alex James – Sr. Principal Architect
Manu Jose – Sr. Manager, Engineering
Igor Braylovskiy – Sr. Director, Engineering

October 2018

Forward-Looking Statements

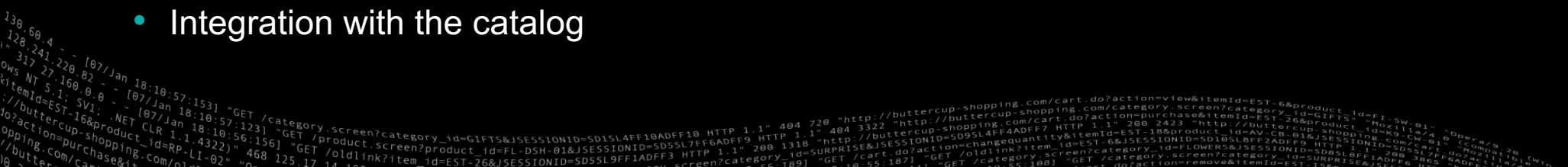
During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results 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

Agenda

- The big picture
  - End to end
  - Architecture
  - Comparing SSC to SplunkD
- Catalog
  - Datasets
  - Rules
- Data Organization
  - Modules
  - Imports
- Spl v2
  - Goals
  - Supported commands
- Search service
  - Running searches
  - Integration with the catalog



End to End

What do you need to do to run a search?

1 → 2 → 3 → 4

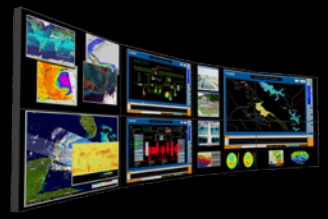
Create index

- Create **index** in **catalog**
- *Route data to index using ingest API*

Create rules & actions

- Create **rules** in **catalog** to enrich sourcetypes to run Regex, Alias, Calc. fields, etc.

Dispatch search



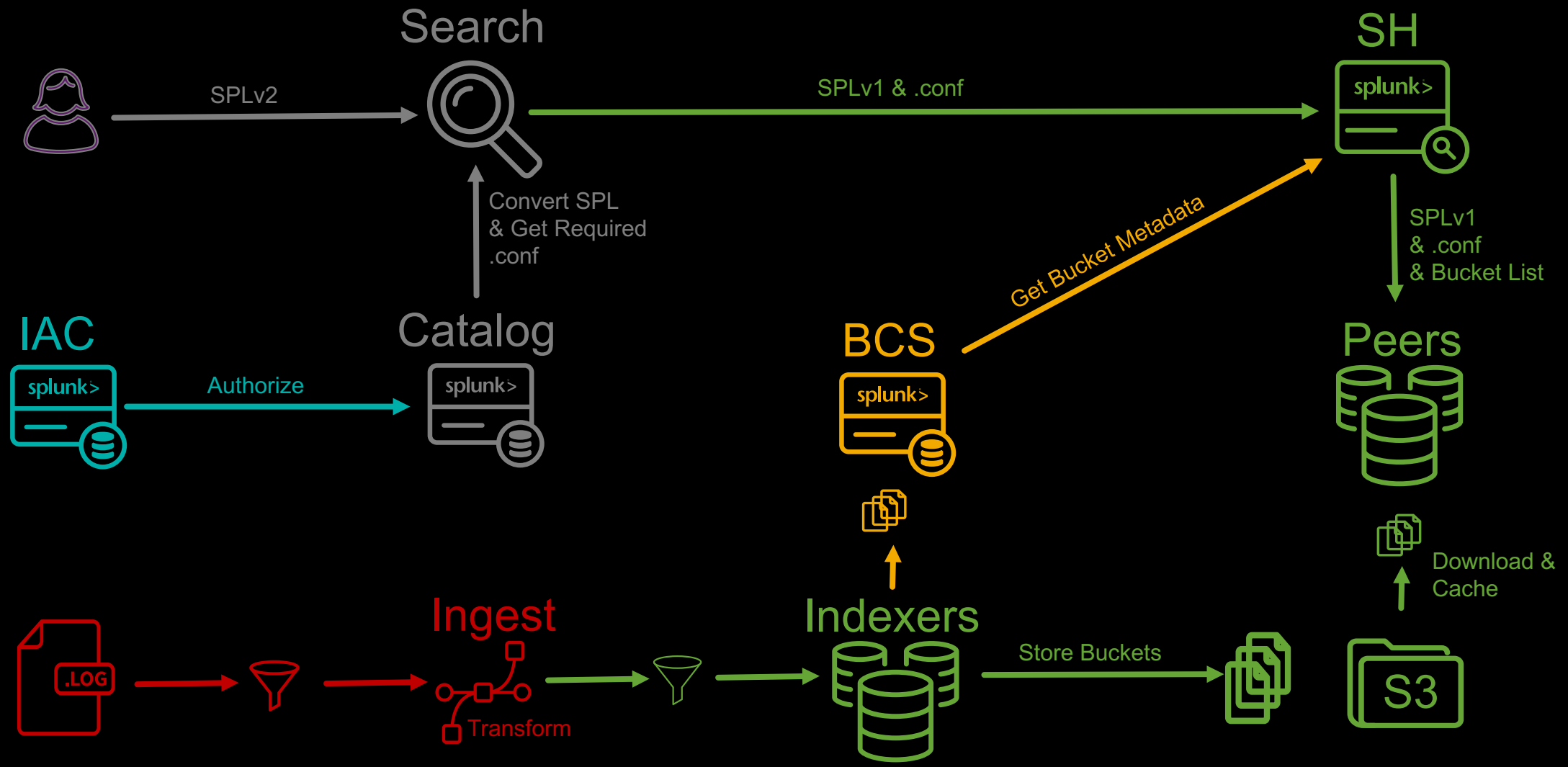
- Dispatch **SPL v2 search** via **Search Service** API call
- Kicks off search in **SplunkD**

Download results

- Query **Search Service** or **Catalog** for job status
- Use **Search Service** to download results

Search Architecture

Architecture Diagram from the Search Perspective



What's Different?

Splunk Enterprise

- Search via embedded Web UI or REST API
- Traditional SPL
- Conf Files
  - Define Indexes, Lookups, Props
- Search & indexing are co-located
- Buckets stored on Indexers
- SplunkD responsible for enforcing Security

SSC

- Search via Search Service or SDK/Apps that use Search Service
- Simplified SPL (v2)
- Catalog
  - Define Datasets and Rules/Actions
- Separated search & indexing
- Buckets stored in S3 and cached
- Catalog and Search Service responsible for enforcing Security

Catalog

Introduction to the new Catalog Service

splunk> .conf18

Catalog

What is the Catalog? What does it store?

- ***The*** source of truth:
  - A REST service that stores metadata about all datasets and rules in Splunk
  - Apps can share metadata via the Catalog
  - Continuously and dynamically updated as Splunk is used
- Datasets are named objects containing data that you can interact with in SPL
  - Examples: indexes, jobs, views, metric indexes, lookups, kvcollections, etc.
  - Dataset metadata stored in the catalog includes:
    - Properties: name, module, kind, owner, createdDate, etc.
    - Type-specific properties: spl for a view, retentionPeriod for an index, etc.
    - Known fields in a dataset
- Rules group a collection of search time enrichments for a SourceType
- Will eventually store relationship metadata and annotations too

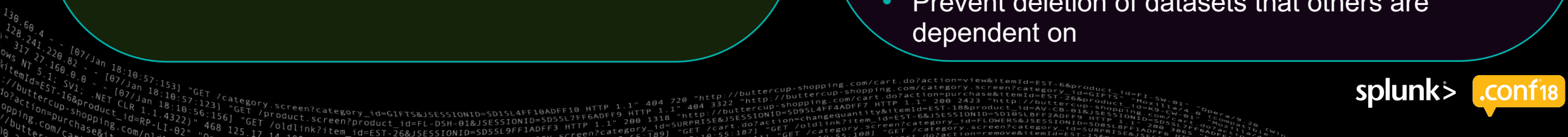
Catalog's Use Cases

Current

- Knowledge object management
  - Indexes, Metrics, Views, KVCollections, Lookups, etc.
- Search history
  - Keep track of all searches run to-date, allows for faster reuse
- Data Discovery
  - Find Data of interest, datasets with specific fields etc.

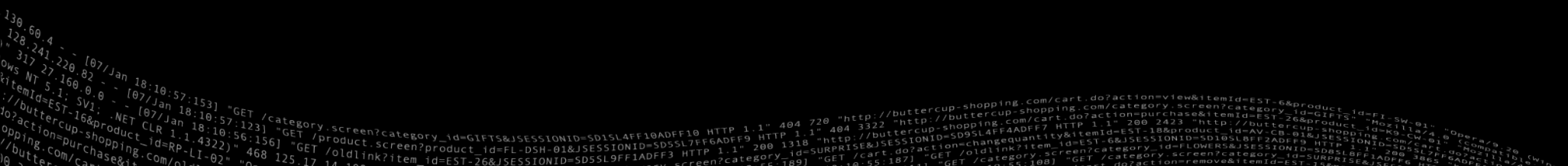
Future

- Read/write metrics catalog
  - Annotate information about metric series
- Cost based optimization
  - e.g optimal order to perform joins
- Recommendations
  - For datasets, across customers
- Data normalization
  - Mapping and publishing of datasets for customers
- Data lineage
  - Annotations that track where fields come from
- Data dependencies
  - Prevent deletion of datasets that others are dependent on



Catalog: Dataset related APIs

- **/<tenant>/catalog/v1/datasets**
  - GET → retrieve information about datasets
  - POST → create a new dataset
- **/<tenant>/catalog/v1/datasets/<resource-name>**
  - GET → retrieve a specific dataset by **dataset-id** or **resource-name**
  - PATCH → update the specific dataset
  - DELETE → delete a specific dataset
- **/<tenant>/catalog/v1/datasets/<resource-name>/fields**
  - GET → retrieve known fields in a specific dataset
  - POST → create (or record the presence of) a new field in a dataset
- **/<tenant>/catalog/v1/datasets/<resource-name>/fields/<name>**
  - GET → retrieve a field in a specific dataset by id or name
  - PATCH → update the specified field
  - DELETE → remove (or delete the record of) the specified field



Catalog

Creating an Index Dataset

POST ~/<tenant>/catalog/v1/datasets
Content-Type: application/json
Accept: application/json
Authorization: <bearerTokenFor(ajames@splunk.com)>

```
{
    "name": "foo",
    "kind": "index",
    "module": "mymodule",
    "disabled": false,
    "frozenTimePeriodInSecs": 188697600,
    "owner": "ajames@splunk.com"
}
```

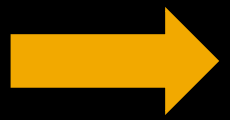
Status: 201 Created
Content-Type: application/json
ContentLength: XXX
Location: /mymodule.foo

```
{
    "name": "foo",
    "kind": "index",
    "module": "mymodule",
    "disabled": false,
    "frozenTimePeriodInSecs": 188697600,
    "owner": "ajames@splunk.com",
    "created": "2018-07-19 06:33:24.000824",
    "modified": "2018-07-19 06:33:24.000824",
    "version": 1,
    "id": "5b50d9740db35a77d54bf32e",
    "createdby": "ajames@splunk.com",
    "modifiedby": "ajames@splunk.com",
    "resourcename": "mymodule.foo",
    "internalname": "mymodule_____foo"
}
```

Catalog: Creating a Field in a Dataset

Example: adding a city field to the foo dataset in mymodule

```
POST ~/<tenant>/catalog/v1/datasets/mymodule.foo/fields
Content-Type: application/json
Accept: application/json
Authorization: <bearerTokenForAJames@splunk.com>
X-Request-ID: <assignedByGatewayService>

{
    "name": "city",
}
```



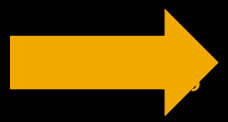
```
Status: 201 Created
Content-Type: application/json
ContentLength: XXX
Location: /city

{
    "name": "city",
    "id": "5b500000db3da700000000",
    "datasetId": "5b50d9740db35a77d54bf32e",
    "dataType": "STRING",
    "fieldType": "UNKNOWN",
    "prevalence": "UNKNOWN",
    "created": " 2018-07-19 06:35:40.000243",
    "modified": "2018-07-19 06:35:40.000243",
    "versionAdded": 2,
    "versionRemoved": null
}
```

Rules and Actions

What are rules and actions?

- **Rules** group **Actions** (or search time enrichments) associated with a SourceType
- **Rules** replace Props.conf

```
[sourcetype::webaccess]
FIELDALIAS-user = uid as user
EXTRACT-errors = device_id=\[w+\](?<err_code>[^:]+)
EVAL-company = collasce(company, "unknown")
LOOKUP-product_info = products ID AS pid \
       OUTPUT Name AS product_name
KV_MODE = auto
```



```
{
    "id": #id#,
    "name": #name#,
    "module": #module#,
    "match": "sourcetype::webaccess",
    "owner": "user”,
    "actions": [
        {
            "id": #id#,
            "kind": "ALIAS",
            "field": "uid",
            "alias": "user”
        },
        {
            "id": #id#,
            "kind": "REGEX",
            "pattern": "device_id=\[w+\](?<err_code>[^:]+)",
            "field": "foo",
            "limit": ?,
            "format": ?,
        }
    ]
}
```

splunk> .conf18

Catalog: Rule related APIs

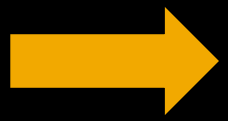
- **/<tenant>/catalog/v1/rules**
  - GET → retrieve information about rules
  - POST → create a new rule
- **/<tenant>/catalog/v1/rules/<rule-id>**
  - GET → retrieve a specific rule by **rule-id**
  - PATCH → update the specific rule
  - DELETE → delete a specific rule
- **/<tenant>/catalog/v1/rules/<rule-id>/actions**
  - GET → retrieve the actions in the specified rule
  - POST → create a new action in the specified rule
- **/<tenant>/catalog/v1/rules/<rule-id>/actions/<action-id>**
  - GET → retrieve the specific action
  - PATCH → update the specified action
  - DELETE → remove the specified action

- ***NOTE:*** *we will also support using a <rule-resourceName> in place of <rule-id> above.*

Catalog: Creating a Rule and nested Action(s)

Example: creating the webaccess rule in mymodule with a nested REGEX action

```
POST ~/<tenant>/catalog/v1/rules
Content-Type: application/json
Accept: application/json
Authorization: <bearerTokenForAJames@splunk.com>
X-Request-ID: <assignedByGatewayService>

{
    "name":"webaccess",
    "module":"mymodule",
    "owner": "ajames@splunk.com",
    "match": "sourcetype::webaccess",
    "actions": [{
        "owner": "ajames@splunk.com",
        "kind": "REGEX",
        "pattern": "[A-Za-z0-9_]+",
        "field": "name"
    }]
}
```



```
Status: 201 Created
Content-Type: application/json
ContentLength: XXX

{
    "name": "webaccess",
    "module": "mymodule",
    "match": "sourcetype::webaccess",
    "owner": "ajames@splunk.com",
    "resourcename": "mymodule.webaccess",
    "created": "2018-07-27 09:14:20.000841",
    "modified": "2018-07-27 09:14:20.000841",
    "version": 1,
    "id": "5b5b8b2c2c0c004d43b14138",
    "createdby": "ajames@splunk.com",
    "modifiedby": "ajames@splunk.com",
    "actions": [{
        "kind": "REGEX",
        "pattern": "[A-Za-z0-9_]+",
        "field": "name",
        "owner": "ajames@splunk.com",
        "created": "2018-07-27 09:14:20.000841",
        "modified": "2018-07-27 09:14:20.000841",
        "version": 1,
        "id": "5b5b8b2c2c0c004d43b14139",
        "createdby": "ajames@splunk.com",
        "modifiedby": "ajames@splunk.com"
```

splunk> .conf18

Modules and Imports

Data Organization

splunk> .conf18

Example Search

mymodule

```
| from main where user="fred"
| stats count() by host
```

Search Service

Catalog Service

catalog

commons-ast

mymodule

- main
- metrics
- users-col
- threats-col
- users (references user-col)
- threats (references threats-col)

X=sourcetype:foo

AL R A C

v1

```
| search index="mymodule_____main" user=CASE("fred")
| stat count by host
```

mymodule ∩ permissions ∩ spl

```
{ needed conf JSON }
```

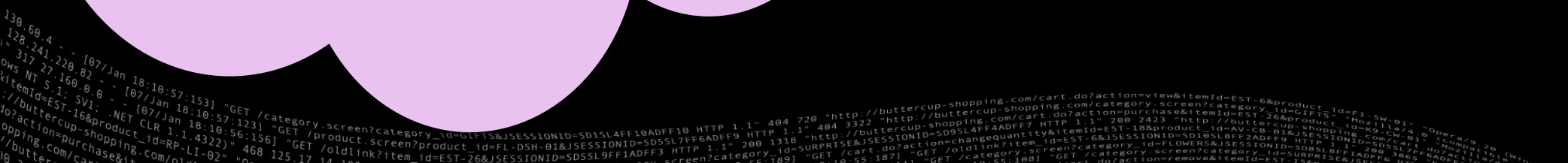
Search Head

conf in Memory

peer1

peer2

peer3

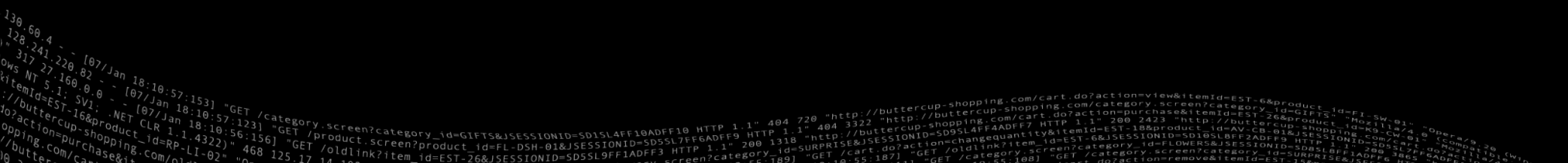


splunk> .conf18

Modules

What is the module thing? And how is it used?

- **Datasets** and **Rules** are organized into **Modules** in the **Catalog**.
  - Every Dataset or Rule has a name and module
  - The concatenation of module + "." + name is a resourceName, it's unique within a tenant
- **Datasets**
  - Datasets are currently referenced by name (not resourceName) in SPL v2
  - Searches are run in the context of a module. Datasets within the search are resolved by their resourceName
  - Dataset in "source" module can be imported into a "target" module
  - The imported dataset can then be referenced by searches executed in the context of the "target" module
- **Rules**
  - Rules group actions (or search time enrichments) that should be run together.
  - Every Rule has a match constraint on sourcetype
  - Rules (and their Actions) are in scope when reading from an index in the same module as the Rule
  - Rules from can be imported into a second module.
  - The imported rule is then in scope for searches run in the second module.



Import Related APIs

- Datasets
  - A dataset is imported by creating an **import dataset**
    - POST ➔ /<tenant>/catalog/v1/datasets
      - **Kind=import** and then set **sourceName** and **sourceModule**
  - Manipulate the import just like any other dataset
    - GET, PATCH, DELETE ➔ /<tenant>/catalog/v1/datasets/<resource-name>
  - You can find all places that import a dataset
    - GET /<tenant>/catalog/v1/datasets/<resource-name>/importedby
- Rules
  - POST /<tenant>/catalog/v1/rules/<resource-name>/importedby
  - GET, PATCH, DELETE /<tenant>/catalog/v1/rules/<import-resource-name>

Catalog: Importing a Dataset

Example: import main index in the shared module into mymodule as localmain

POST ~/<tenant>/catalog/v1/datasets

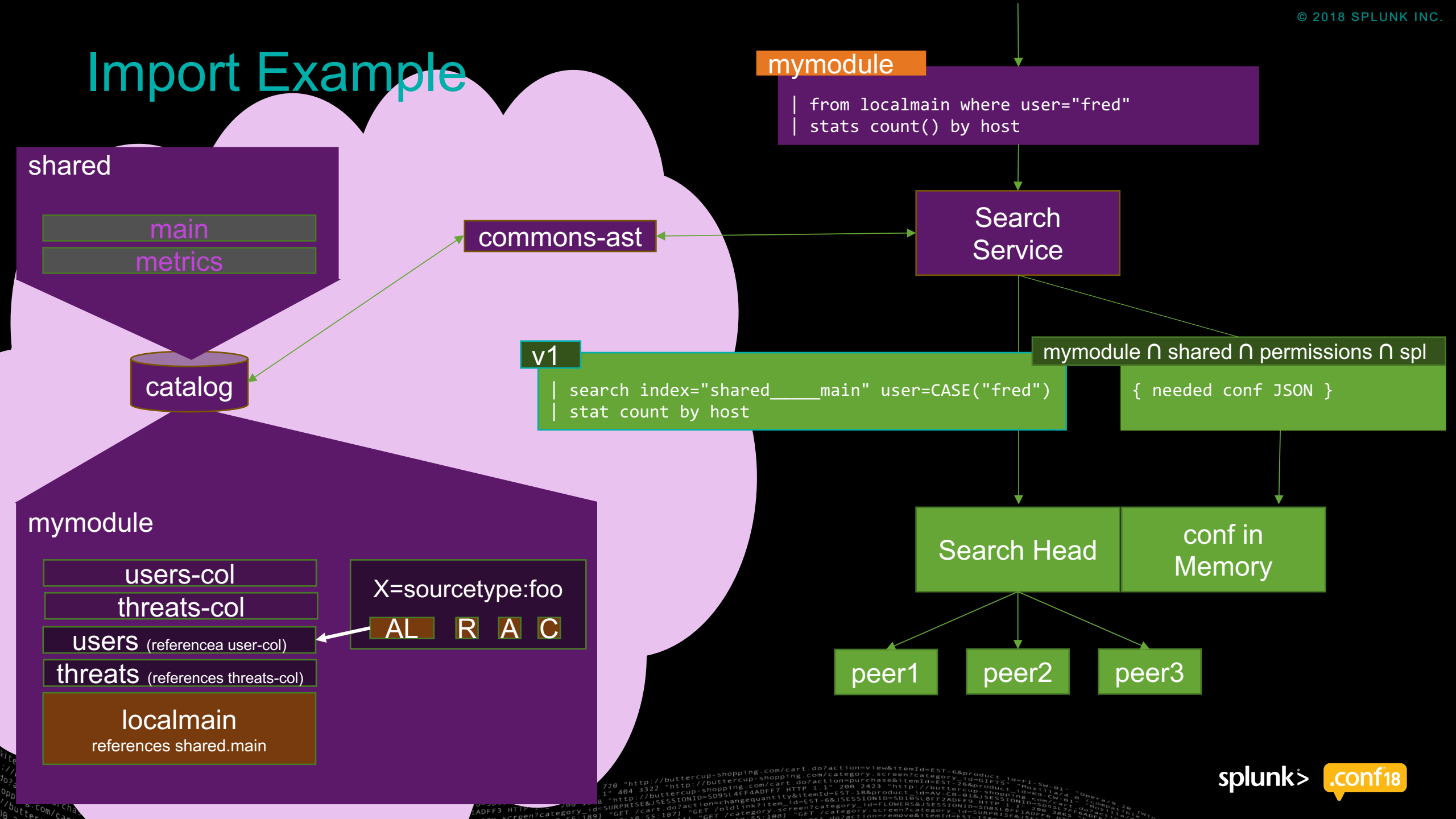
```
Content-Type: application/json
Accept: application/json
Authorization: <bearerTokenFor(ajames@splunk.com)>

{
    "name": "localmain",
    "module": "mymodule",
    "kind": "import",
    "sourceName": "main",
    "sourceModule": "shared"
}
```

```
Status: 201 Created
Content-Type: application/json
ContentLength: XXX
Location: ../mymodule.localmain

{
    "name": "localmain",
    "module": "mymodule",
    "kind": "import",
    "sourceName": "main",
    "sourceModule": "shared",
    "id": ”0001d9740db35a77d54bf32e",
    "owner": "ajames@splunk.com",
    "created": "2018-07-19 06:33:24.000824",
    "modified": "2018-07-19 06:33:24.000824",
    "version": 1,
    "createdby": "ajames@splunk.com",
    "modifiedby": "ajames@splunk.com",
    "resourcename": "mymodule.localmain",
    "internalname": null,
     "version": 1
}
```

GET ~/<tenant>/catalog/v1/datasets/mymodule.localmain

splunk> .conf18

Import Example



SPL v2

Simplified and Consistent SPL

splunk> .conf18

Spl v2

Why? And what is it?

- Goals
  - Taking the best of Traditional Spl & applying lessons learned.
  - Syntax Consistency & Formal Grammar
  - Less commands to learn
- Key commands (so far)
  - **from**, **into**
  - **search**, **where**
  - **eval**, **rex**, **mvexpand**, **rename**
  - **bin, stats**, **chart**, **timechart**
  - **union**, **join**, **lookup**
  - **head**, **sort**, **dedup**

Spl v2: From

The most important command in SPL v2, for reading data 'from' a dataset

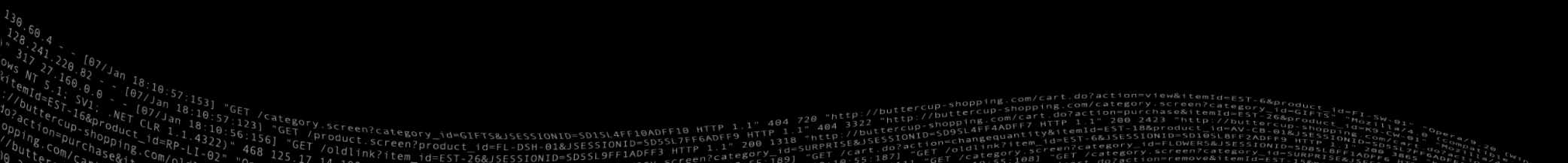
- SSC supports more clauses:
  - `| from <dataset> where <filter> groupby <field-list> select <projection-list> orderby <field-list>`
- SSC supports more types of datasets:
  - `| from index:main` ➔ `search index=main`
  - `| from job:<sid>` ➔ `| loadjob <sid>`
  - `| from lookup:users` ➔ `| inputlookup users`
  - `| from metric:metrics where groupby host select host, avg(cpu.percentage)` ➔ `| mstats avg(cpu.percentage) by host where metric_name="cpu.percentage"`
  - `| from view:mysearch` ➔ `<spl-defined-in-mysearch>`
  - `| from [<spl>]` ➔ `<spl>`
- No need to specify type:
  - i.e. `| from ~~type:~~name` works
  - Dataset names are unique (within a module), enforced by the catalog.
  - Specifying type is now essentially an assertion.
  - Best to avoid specifying type.

splunk> .conf18

Spl v2

Other commands

- Syntax changes only:
  - **apply**, **bin**, **chart**, **eval**, **fields**, **fit**, **head**, **join**, **lookup**, **mvexpand**, **rename**, **rex**, **sort**, **search**, **timechart**, **union**, **where**
- Syntax & Semantics changes:
  - **dedup**: no sorting option, use **sort** instead
  - **stats**: now supports binning on time, i.e. | stats count by host, _time span=1m
- Alternatives to existing commands:
  - `mstats` ➔ `| from metric-index`
  - `mcatalog` ➔ `| from catalog.metrics …`
  - `inputlookup` ➔ `| from lookup`
  - `outputlookup` ➔ `| into mode=replace lookup`
  - `sendalert` ➔ `| into alerts`
- Coming Soon:
  - **fieldsummary**, **tstats** (or equivalent), **transaction (**simplified)
  - Let us know what commands you really really need…



splunk> .conf18

Search Service: Dispatching a Search

```
POST ~/<tenant>/search/v1/jobs
Content-Type: application/json
Accept: application/json
Authorization: <bearerTokenFor(ajames@splunk.com)>

{
    "module": "mymodule",
    "query":"| from index:metrics select avg(CPU)"
    "extractAllFields": false,    // default
    "maxTime": 3600,              // 1hr default
    "timeOfSearch": now(),        // default is now
    "timeFormat": "%FT%T.%Q%:z", // default
    "ttl":86400                   // i.e. 24hrs default
    "parameters": {
        "earliest": "-24h@h",
        "latest": "now()",
    }
}
```

```
Status: 201 Created
Content-Type: application/json
ContentLength: XXX

{
    "module": "mymodule",
    "query":"| from index:metrics select avg(CPU)"
    "sid": "1132142.2”,
    "status": "running",
    "extractAllFields": false,
    "maxTime": 3600,
    "timeOfSearch": "2018-07-27 09:14:20.000841",
    "resultsAvailable": 0,
    "timeFormat": "%FT%T.%Q%:z",
    "ttl":86400 ,
    "percentComplete": 0,
    "messages":[],
    "parameters": {
        "earliest": "-24h@h",
        "latest": "now()",
    }
}
```

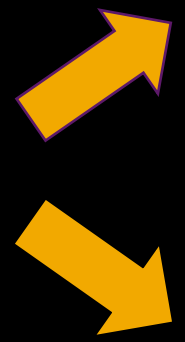
Search Service: Canceling a search

```
PATCH ~/<tenant>/search/v1/jobs/1132142.2
Content-Type: application/json
Accept: application/json
Authorization: <bearerTokenFor(ajames@splunk.com)>
X-Request-ID: <assignedByGatewayService>

{
    "status" : "cancelled"
}
```

```
Status: 200 OK
Content-Type: application/json
ContentLength: XXX

{
    "module": "mymodule",
    "query":"| from index:metrics select avg(CPU)",
    "sid": "1132142.2",
    "status": "cancelled",
    "extractAllFields": false,
    "maxTime": 3600,
    "timeOfSearch": "2018-07-27 09:14:20.000841",
    "resultsAvailable": 0,
    "timeFormat": "%FT%T.%Q%:z",
    "ttl":86400 ,
    "percentComplete": 100,
    "messages":[],
    "parameters": {
        "earliest": "-24h@h",
        "latest": "now()",
    }
}
```

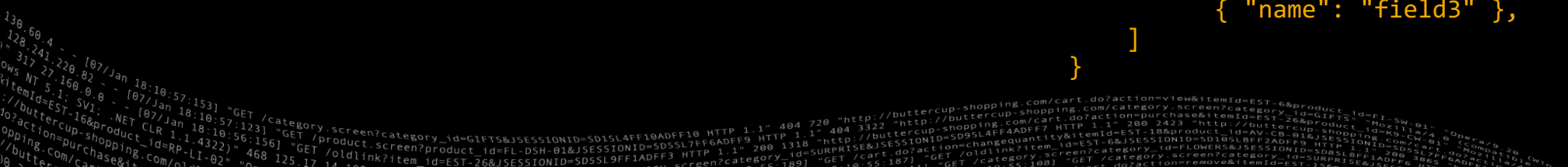
Search Service: Download Results

```
GET ~/<tenant>/search/v1/jobs/1132142.2/results
Accept: application/json
Authorization: <bearerTokenFor(ajames@splunk.com)>
X-Request-ID: <assignedByGatewayService>
```



```
Status: 200 OK
Content-Type: application/json
ContentLength: XXX

{
    "nextLink": "/results",
    "wait": "2000"
}
```

```
Status: 200 OK
Content-Type: application/json
ContentLength: XXX

{
    "messages": [ <array of messages>],
    "results": [{
            "field1": "value1",
            "field2": "value2",
            "field3": "value3"
    }],
    "fields": [
            { "name": "field1" },
            { "name": "field2" },
            { "name": "field3" },
    ]
}
```



splunk> .conf18

Search Service: Integration with the Catalog

- When a search is dispatched:
  - A **Job** dataset is created in the Catalog
  - **The Job** dataset is *not* updated until Job is **finished**
  - Latest state is accessible via Search Service
  - When Search finishes the **Job** Dataset is updated
    - Initially just changing state and duration etc.
    - Eventually this will include more statistics.
- **Job** Dataset can stay in Catalog forever if necessary
  - We will make **| from <sid>** will work beyond the ttl

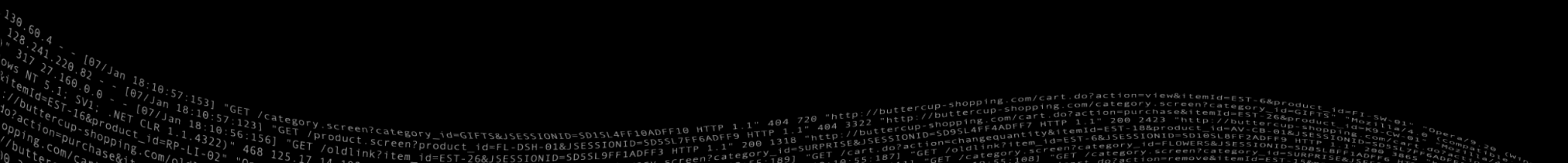
```
{
"id": #id#
"name": "sid.123141424",
"status": "running“
"module": "mymodule",
"kind": "job",
"sid": "123141424",
"query": "| from indexname | stats count() by host",
"timeOfSearch": "2018-07-27 09:14:20.000841",
"resultsAvailable": 0,
"spl": "search index=mymodule_____indexname |
                stats count by host“
"timeFormat": "%FT%T.%Q%:z",
"deleteTime": "2018-07-28 09:14:20.000841“,
"parameters": {
     "earliest": "-24h@h",
     "latest": "now()",
 },
"owner": #jobexecutor#,
"created": #createdTime#,
"modified": "#modifiedTime#,
"version": 1,
"createdby": #jobexecutor#,
"modifiedby": #jobexecutor#
}
```

splunk> .conf18

Six .conf 2018 Sessions on Splunk Developer Cloud

- Building Apps for Splunk Developer Cloud DEV1902
- Splunk Developer Cloud Services and Features DEV1552
- Developer Tools for Splunk Developer Cloud DEV1841
- Partners Build Apps on Splunk Developer Cloud DEV1846
- Dashboards and Analysis UI Components for Developers DEV1703
- Intro to SPLv2, the Module System and the Catalaog DEV2043

Sign up at splunk.com/projects or splunk.com/sdc



splunk> .conf18

Questions?