COMMUNITY
THE ASF CONFERENCE
CODE

Secure Apache Spark on Kubernetes with Apache Ranger

Chaoran Yu & Claudia Sun

Agenda

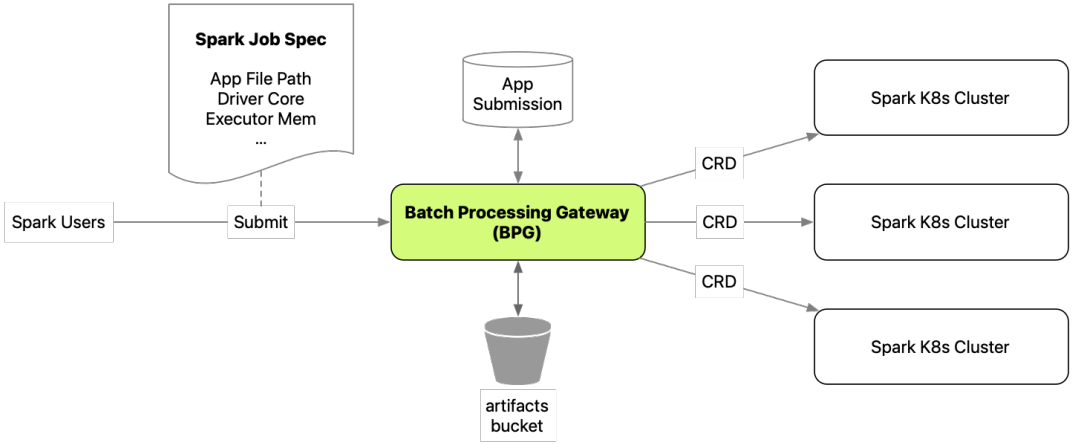
- Apache Spark on Kubernetes
- Security Challenges for a Apache Spark-based Batch Platform
- Apache Ranger Plugin for Queue Authorization
- Apache Ranger Plugin for Spark SQL
    - History
    - HMS + Hive Table
    - IRC + Iceberg Table
- Possible Next Steps

Apache Spark on Kubernetes

- Apache Spark
  - De-facto standard for batch data processing and analytics
- Running Apache Spark in the Cloud
  - Apache Spark on Kubernetes mode
  - Ease of use, multi-tenancy, resource management, security

Batch Processing Gateway

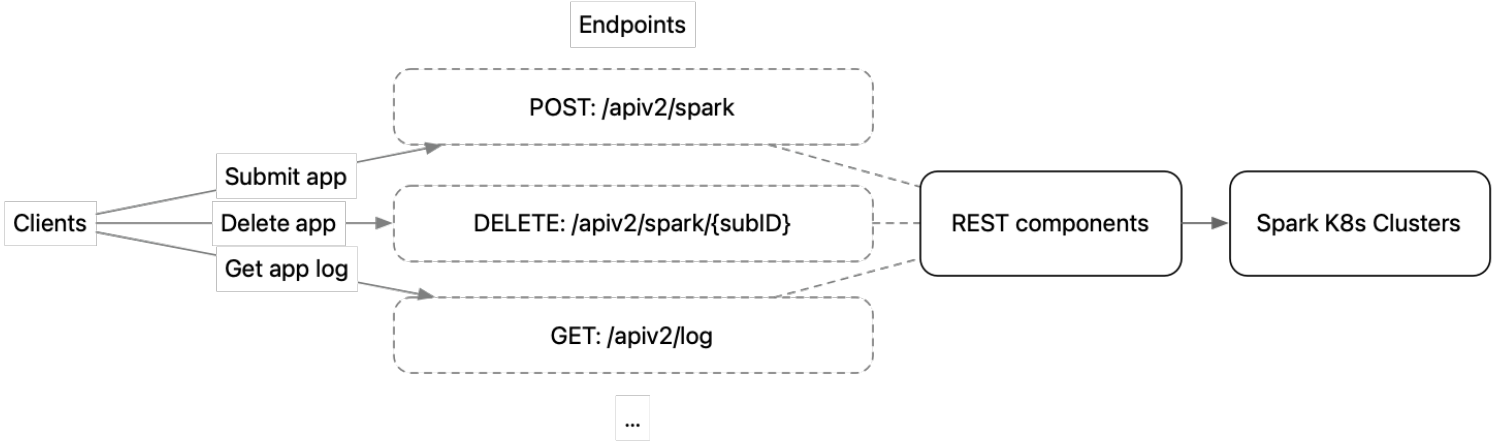
https://github.com/apple/batch-processing-gateway



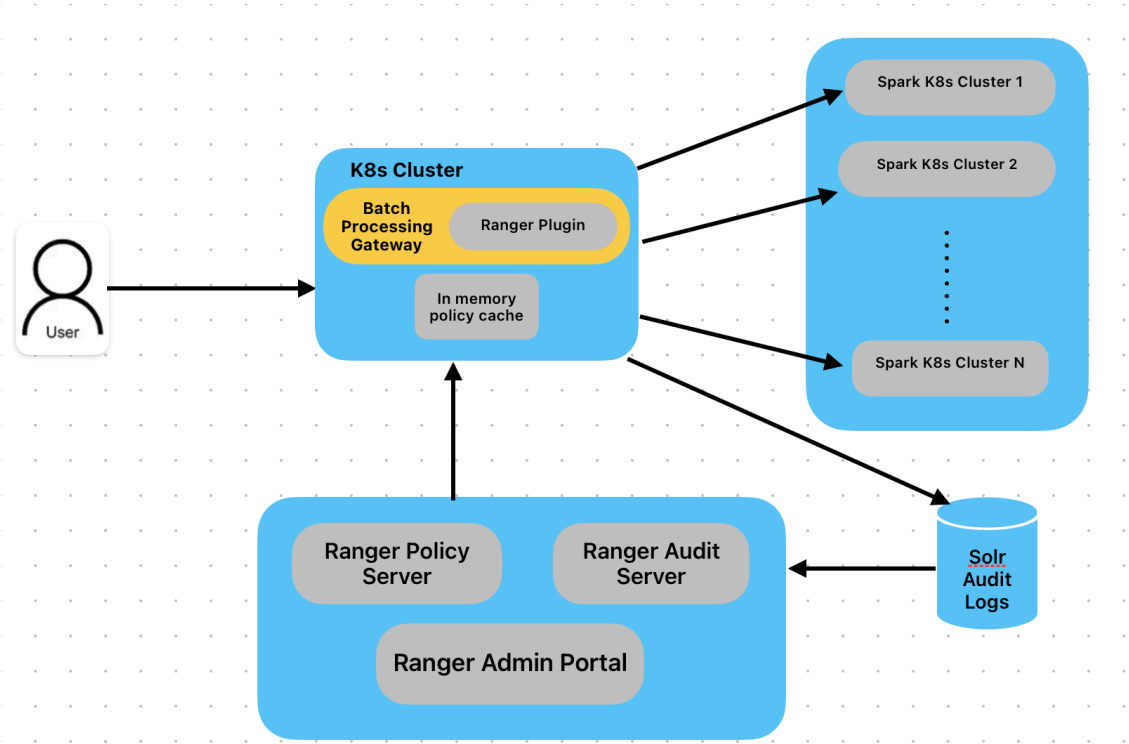
Security Challenges

- Authorization
  - Who is authorized to perform actions on a certain API endpoint (e.g. submit, status, log)
  - Who is authorized to access a certain dataset (i.e. data access control)
- Authentication

API Authorization

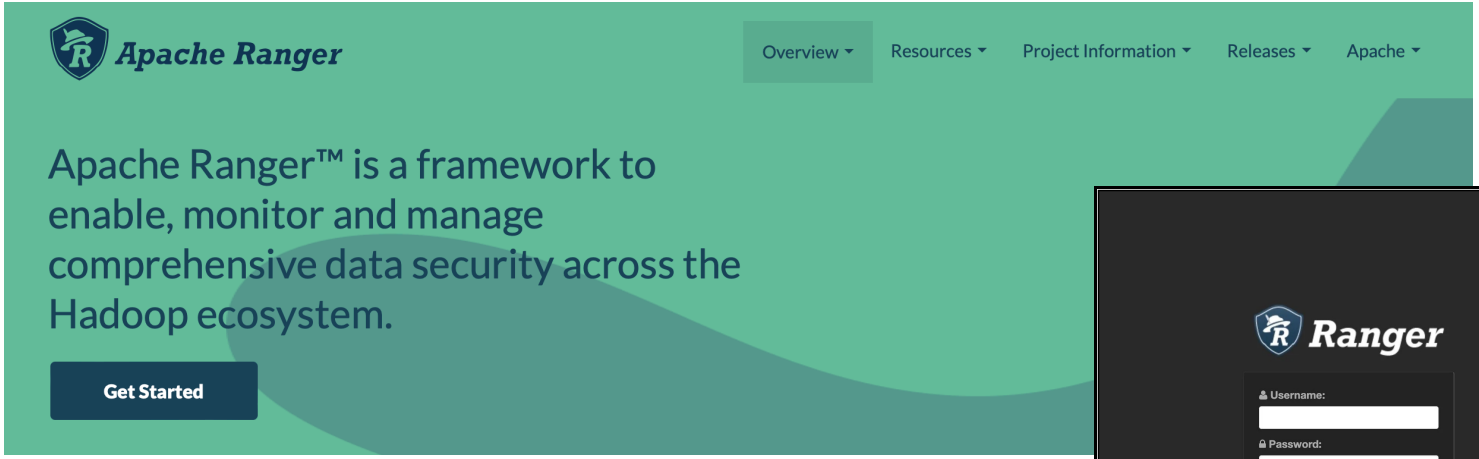


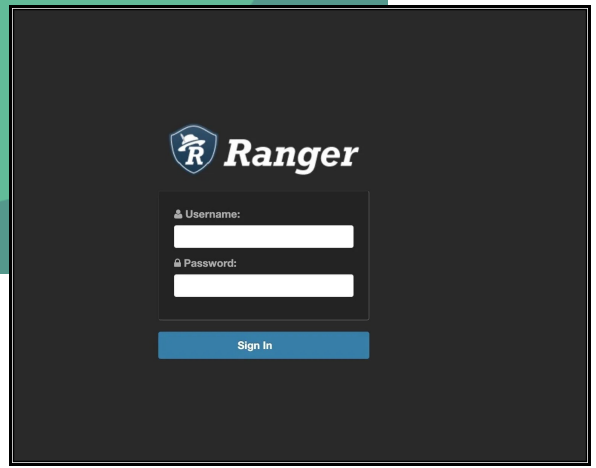
API Authorization Architecture



https://github.com/apple/batch-processi

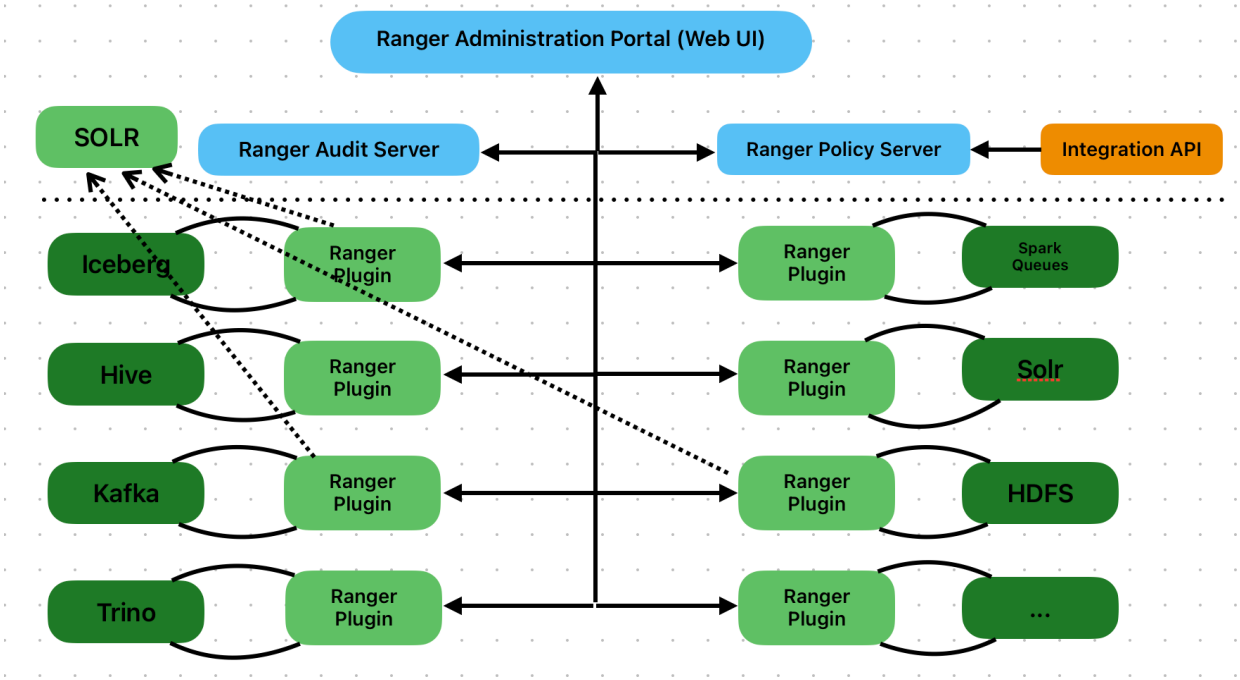
Apache Ranger





https://ranger.apache.org/

Architecture



Apache Spark Queues Authorization with Apache Ranger

Policy Creation

Ranger | Access Manager | Audit | Security Zone | Settings | admin

Service Manager > spark-queue Policies > Create Policy

Create Policy

Policy Details :

Policy Type: Access

Add Validity Period

Policy Name *: enabled | normal

Policy Label: Policy Label

Spark Queue *: recursive

Description

Audit Logging: YES

Allow Conditions :

Select Role	Select Group	Select User	Permissions	Delegate Admin	
Select Roles	Select Groups	Select Users	Add Permissions +		×

+

Exclude from Allow Conditions :

Select Role	Select Group	Select User	Permissions	Delegate Admin	
Select Roles	Select Groups	Select Users	Add Permissions +		×

+

Apache Spark Queue Authorization with Ranger

Apache Solr Auditor

Ranger | Access Manager | Audit | Security Zone | Settings | rangerusersync

Access | Admin | Login Sessions | Plugins | Plugin Status | User Sync

START DATE: 09/03/2024

Exclude Service Users :

Entries : 0 | Last Updated Time : 09/02/2024 08:37:31 PM

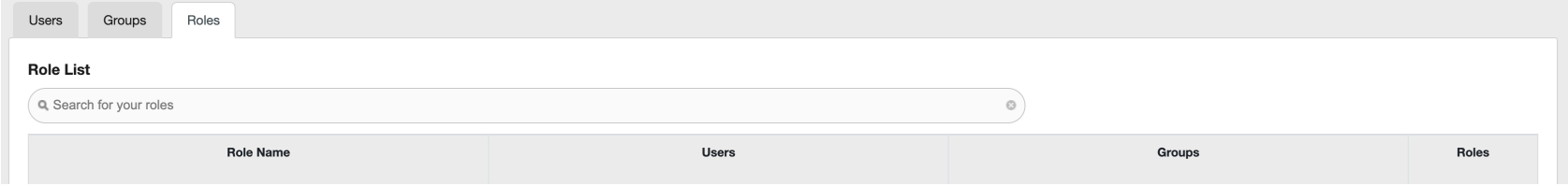
Policy ID	Policy Version	Event Time	Application	User	Service Name / Type	Resource Name / Type	Access Type	Permission	Result	Access Enforcer	Agent Host Name	Client IP	Cluster Name	Zone Name	Event Count	Tags
No Access Audit found!																
89	29	11/14/2023 12:22:28 PM	skate-spark-queue-authorizer	raimldpi	spark-queue spark queue	poc_02 queue		log	Allowed	ranger-acl	674f22efca11					
--	--	11/14/2023 12:22:28 PM	skate-spark-queue-authorizer	user	spark-queue spark queue	poc_02 queue		log	Denied	ranger-acl	674f22efca11					

All the audits related to any Ranger plugin will be synced to Solr and will be visible here on the UI

Dataset Authorization

Role-based access control (RBAC), Attribute-based access control (ABAC), and custom data resources/types in Ranger.

- Apache Spark Queues
- Apache Hive Tables
- Apache Iceberg Tables



Apache Ranger Spark Plugin Integration
History

- Started as a small solo-dev OSS project
  - https://github.com/yaooqinn/spark-ranger
- Author *really* tried to get it into Apache Ranger mainline (2018) ❌
  - https://issues.apache.org/jira/browse/RANGER-2128
- Integrated with Apache Submarine project (2020)

Apache Ranger Spark Plugin Integration
History Cont.

- *Retired* from Apache Submarine Nov. 2021 😔
  - https://github.com/apache/submarine/pull/796
- Moving to Apache Kyuubi Project as a Spark security module 🤗 But no Iceberg and Datasource V2 support😔
- Add Data Source V2 support, specifically for Iceberg support in Ranger Plugin for many of the core Spark actions. Supports Spark 3.2, 3.3 & 3.4(currently)

Apache Ranger Iceberg Service Creation

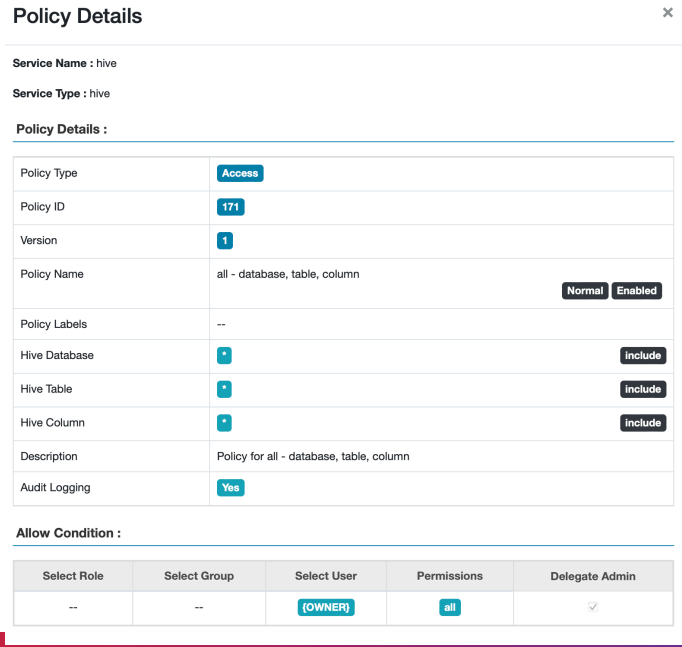
- **Ranger Service Definition**

```
{ "name": "Iceberg",
  "resources": [ {
    "itemId": 1,
    "name": "Table",
    "type": "string",
    "matcher":"org.apache.ranger.plugin.resourcematcher.RangerDefaultResourceMatcher",
    "accessTypes":["CREATE", "UPDATE", "INSERT", "DELETE"...]
    }, ...]
```

Apache Ranger Spark Plugin Integration

Ranger Spark Plugin supports three levels of access control:

- Namespace/Database
- Table
- Column



Policy Details

Service Name : hive

Service Type : hive

Policy Details :

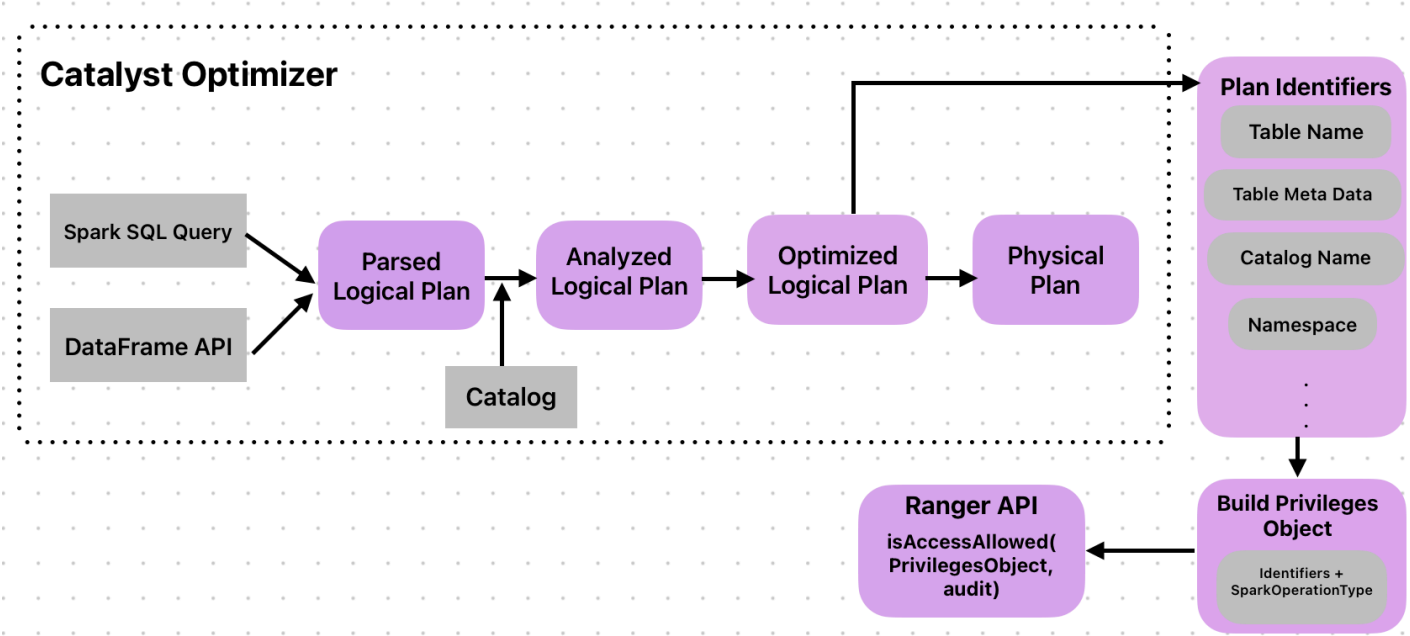
Policy Type	Access
Policy ID	171
Version	1
Policy Name	all - database, table, column Normal Enabled
Policy Labels	--
Hive Database	* include
Hive Table	* include
Hive Column	* include
Description	Policy for all - database, table, column
Audit Logging	Yes

Allow Condition :

Select Role	Select Group	Select User	Permissions	Delegate Admin
--	--	{OWNER}	all	✓

Apache Ranger Spark Plugin Integration

Catalyst Optimizer



Apache Ranger Plugin Integration

- Hive Meta Store + Hive Tables
    - sparkSession.sessionState.catalog
- Simple plan for a count on a Hive table

Aggregate [value#153], [sum(cast(key#152 as bigint)) AS k1#150L, value#153 AS v1#151]
 +- **HiveTableRelation** [`default`.`src`, parquet, Data Cols: [key#152, value#153], Partition Cols: []]

Simplified further as *Aggregate->Filter->HiveTableRelation*

Apache Ranger Plugin Integration

- The Ranger Spark plugin *injects* itself via RangerSparkAuthzExtension API into Catalyst's plan optimizer pipeline
- It evaluates the Plan Nodes (*Aggregate, Filter, HiveTableRelation, etc.)* and **maps** them to Ranger *actions* (QUERY, UPDATE, ALTER, etc.)
- If policy permits all extracted Ranger actions, the query is allowed to continue.
- Otherwise, you get

  21/12/03 10:07:50 ERROR SparkRangerAuthorizationExtension: *** Permission denied: user [alice] does not have [SELECT] privilege on [default/src/key] ***

Apache Ranger Plugin Integration

- Iceberg REST Catalog + Iceberg Tables

Instead of reading from Spark Session Catalog, Ranger Plugin needs to cope with Spark SQL Catalog to get catalog information.

- **sparkSession.sessionState.catalogManager.currentCatalog**

This is due to different Catalog Mechanism. Iceberg does not use the Hive Metastore for storing its metadata. Instead, it uses its own catalog, which can be accessed through a REST API or other mechanisms.

Apache Ranger Plugin Integration

Apache Iceberg tables are designed to work specifically with the **DataSource V2** API in Spark, which was introduced in Spark 2.3

Examples:

ShowTables Command's logical plan in DataSource V1:

```
+- ShowTablesCommand default, [namespace#16, tableName#17, isTemporary#18], false
```

ShowTables Command's logical plan in DataSource V2:

```
+- ShowTables [namespace#16, tableName#17, isTemporary#18]
            +- ResolvedNamespace V2SessionCatalog(spark_catalog), [default]
```

Possible Next Steps

- Unified policy support (think: Apache Spark, Trino, et al. share same policy descriptor from Apache Ranger)
- Security hardening (JVM agent, honoring policy against bare file paths, strong principal identity verification)

Thanks For Attending
Q&A