DATA VIRTUALITY MASTERCLASS

Topic: Logging with log4j

What to expect from this session?

- This track will tell you how to learn more about the various logging capabilities of Data Virtuality using log4j and how to send these logs to other systems.
- This is useful and recommended as per International CyberSecurity Standards that the logs are stored on systems where the admins of the system administrators don't have access to.
- Example: Logs sent to IT Security Department with read only access to the Internal IT Audit function.

data virtuality

Logfiles

Logfiles

When analyzing the behavior of Data Virtuality Server in order to trace activities of the server or perform debugging, it is recommended to have a look at the log files in the Data Virtuality Server's directory.

- There is one log file that contains all the information from the last server startup (boot.log)
- And another file that writes all queries and errors (server.log)

Logfiles Location

Linux:

%pathToDVserver%/standalone/log/boot.log

(per Best Practice: /opt/datavirtuality/dvserver/standalone/log/boot.log)

%pathToDVserver%/standalone/log/server.log

(per Best Practice: /opt/datavirtuality/dvserver/standalone/log/server.log)

Windows:

%pathToDVserver%\standalone\log\boot.log

(per Best Practice: C:\Program Files(x86)\datavirtuality\dvserver\standalone\log\boot.log)

%pathToDVserver%\standalone\log\boot.log

(per Best Practice: C:\Program Files(x86)\datavirtuality\dvserver\standalone\log\boot.log)

boot.log

Some Information you can get from the boot.log file(s) are:

- The connections used for the configuration databases, if different from H2
- JAVA_HOME system environment variable that is used
- Java runtime which the server runs on
- The machine's hostname and Fully Qualified Domain Name (FQDN)
- Several important directories that are being used
- Memory settings of the Java Virtual Machine
- More on our Official Documentation

server.log

The server.log contains a treasure trove of information such as but not limited to:

- System actions performed by the Data Virtuality Server
- All commands that were sent to the server
- All errors that occurred
- Connection timeouts
- License Information
- Much More

server.log

- Our Platform uses SQL statements for all operations
- Track all the modifications that might have been made to the server options, optimization jobs or additional flags that were used when a statement was executed.
- The major application of the log is troubleshooting.
- While Data Virtuality Studio only gives an error message after a failed query run, the error's stack trace can be read in the server.log

data virtuality

Stored Procedures Used for Logging

SYSADMIN.logMsg

This procedure logs a message to the underlying logging system.

SYSADMIN.logMsg(OUT logged boolean NOT NULL RESULT, IN level string NOT NULL DEFAULT 'DEBUG', IN context string NOT NULL DEFAULT 'org.teiid.PROCESSOR', IN msg object NOT NULL)

If the message has been logged, it returns TRUE. The level can be one of the Log4j levels: OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE. The level defaults to DEBUG and the context defaults to org.teiid.PROCESSOR.

SYSADMIN.logMsg (Example)

```
CALL SYSADMIN.logMsg(msg => 'some debug', context => 'org.something')
```

This will log the message 'some debug' at the default level DEBUG to the context org.something.

SYSADMIN.isLoggable

This procedure checks if logging is enabled at the given level and context.

SYSADMIN.isLoggable(OUT loggable boolean NOT NULL RESULT, IN level string NOT NULL DEFAULT 'DEBUG',
IN context string NOT NULL DEFAULT 'org.teiid.PROCESSOR')

```
call "SYSADMIN.isLoggable"(
    "level" => 'DEBUG'/* Mandatory */,
    "context" => 'org.teiid.PROCESSOR'/* Mandatory */
);;
```

If logging is enabled, it returns TRUE. The level can be one of the Log4j levels: OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE.

The level defaults to DEBUG and the context defaults to org.teiid.PROCESSOR.

SYSADMIN.isLoggable (Example)

```
begin
if ((select loggable from "SYSADMIN.isLoggable"("context" => 'org.teiid.PROCESSOR')))
begin
call "SYSADMIN.logMsg"(
    "level" => 'DEBUG'/* Mandatory */,
    "context" => 'org.teiid.PROCESSOR'/* Mandatory */,
    "msg" => 'Testing Logging'/* Mandatory */
);
end
end
```

data virtuality

Reading & Parsing Logs

Parsing logs locally via Otros LogViewer

- OtrosLogViewer is a great tool for working with Data Virtuality Server log files.
- Download: Releases · otros-systems/otroslogviewer · GitHub
- Unzip and start, no installation needed.
- The pattern used in Data Virtuality Server log file needs to be announced in "Log4j pattern parser editor", available via "Tools" -> "Show Log4j pattern parser editor". The pattern should be added before any logfile is opened otherwise the OtrosLogViewer will run into memory problems when trying to parse the log.
- Pattern needed for Data Virtuality Server log files:
- type=log4j

  pattern=TIMESTAMP LEVEL LOGGER (THREAD) MESSAGE

  dateFormat=HH:mm:ss,SSS
- Keep in mind that this application might suffer for large logs files when no sufficient RAM is available.

Cloud based log parsing without Cloudwatch Agent

data virtuality

- Parse application logs via Athena and Glue catalog
  - Used when you don’t want to install the Cloudwatch-agent for some reason
  - we can also use Elasticsearch ingest data to Elasticsearch and use ELK stack using logstash as logstash can read GELF format and is good at multiline log files.
  - We can also use Serde (Serialization/ Deserialization) libraries on Athena using log4j (Grok SerDe)
  - The Logstash Grok SerDe is a library with a set of specialized patterns for deserialization of unstructured text data, usually logs.
  - Each Grok pattern is a named regular expression.
  - You can identify and re-use these deserialization patterns as needed.
  - This makes it easier to use Grok compared with using regular expressions.
  - Grok provides a set of pre-defined patterns.
  - You can also create custom patterns.

GrokSerDe

- To specify the Grok SerDe when creating a table in Athena, use the ROW FORMAT SERDE 'com.amazonaws.glue.serde.GrokSerDe' clause, followed by the WITH SERDEPROPERTIES clause that specifies the patterns to match in your data, where:
- The input.format expression defines the patterns to match in the data. (Mandatory)
- The input.grokCustomPatterns expression defines a named custom pattern, which you can subsequently use within the input.format expression. (optional).
- To include multiple pattern entries into the input.grokCustomPatterns expression, use the newline escape character (\n) to separate them, as follows: *'input.grokCustomPatterns'='INSIDE_QS ([^\"]*) \n INSIDE_BRACKETS ([^\\]]*)').*
- The STORED AS INPUT FORMAT and OUTPUTFORMAT clauses are required.
- The LOCATION clause specifies an Amazon S3 bucket, which can contain multiple data objects.
  - All data objects in the bucket are deserialized to create the table.
  - https://docs.aws.amazon.com/athena/latest/ug/grok-serde.html

Sending the logs to AWS CloudWatch using agent

- AWS CloudWatch allows you to collect logs from your AWS EC2 instances.
- To send the application or server logs to AWS CloudWatch, we need to install the CloudWatch agent on the EC2 instance running DV Server.
- Steps Needed:
  - Appropriate role to be attached to the instance to communicate with AWS CloudWatch
  - AWS CloudWatch agent installation
  - Configuration of the AWS CloudWatch agent
  - Testing logs in AWS CloudWatch portal

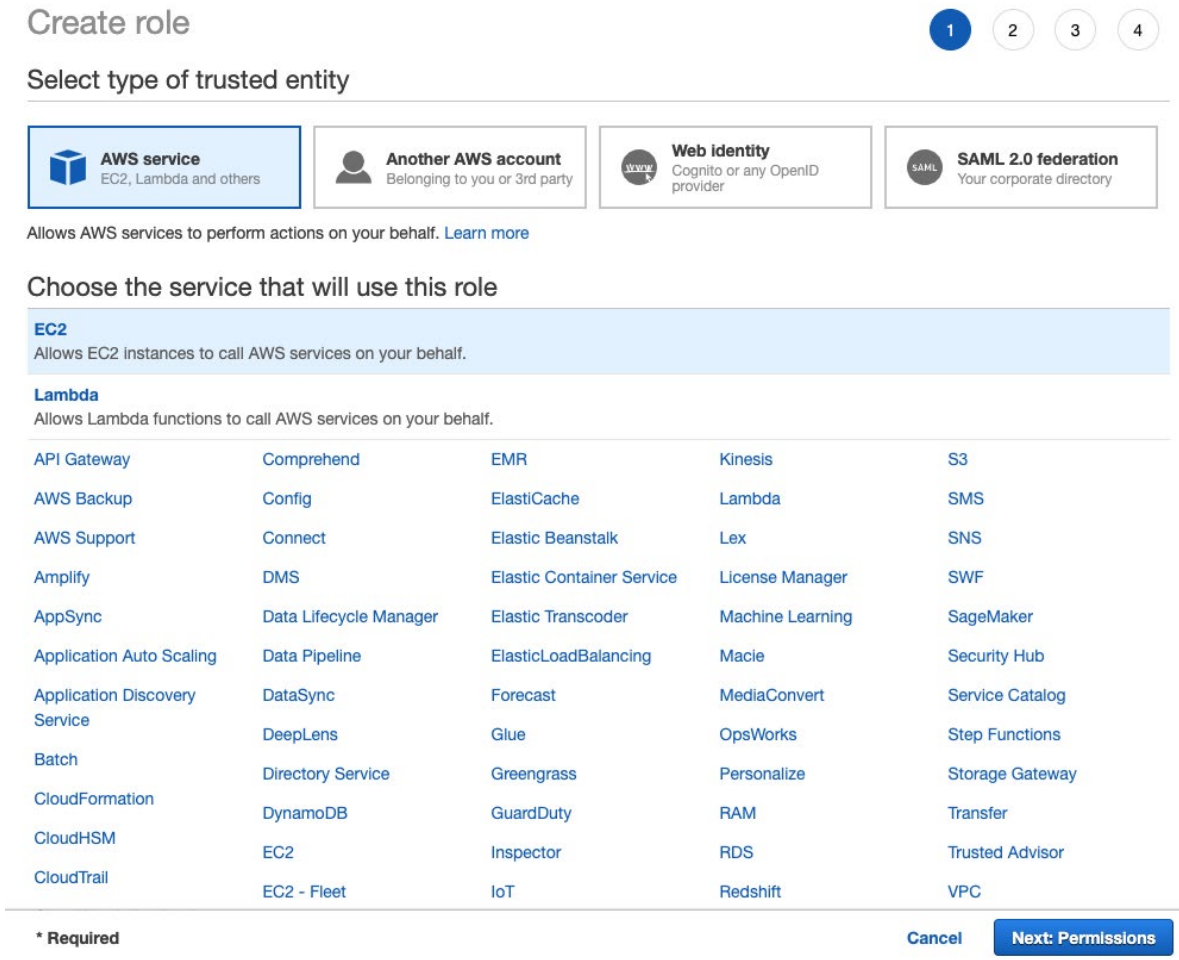
Sending the logs to AWS CloudWatch

- IAM Role
  - In order to send the server logs to AWS CloudWatch, we need to attach a role to the EC2 instance with appropriate permissions.
  - The role will allow making changes in the AWS CloudWatch.
  - It should contain the below policy
    - CloudWatchAgentServerPolicy

Create a New IAM Role

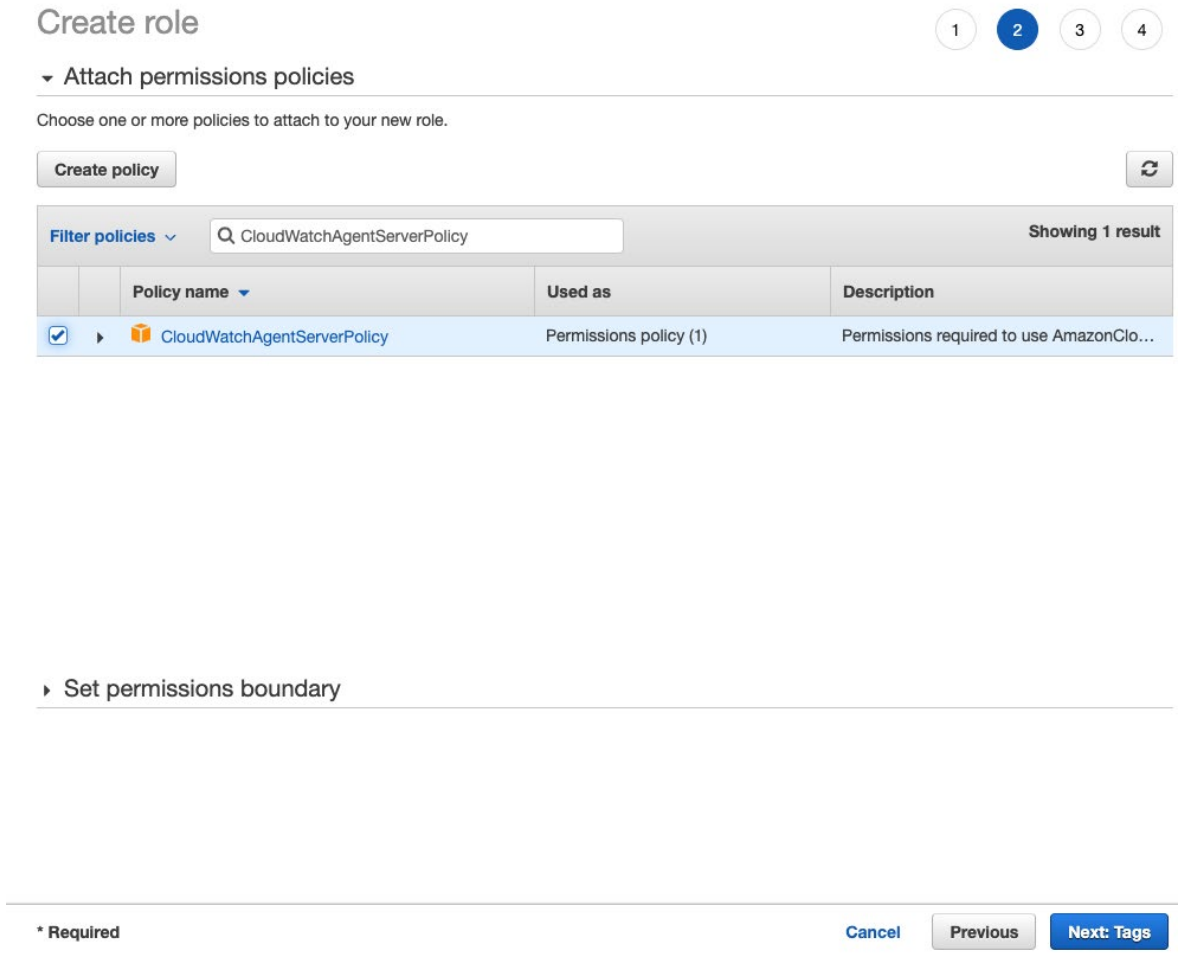
To allow an EC2 instance to communicate with CloudWatch, you first need to create an IAM Role. You only need to do this once, so you can skip this step after it's been created.

- Open the IAM console. From the menu, select Roles and then click the Create role button.
- Under Choose the service that will use this role, select EC2 and click the Next: Permissions button:



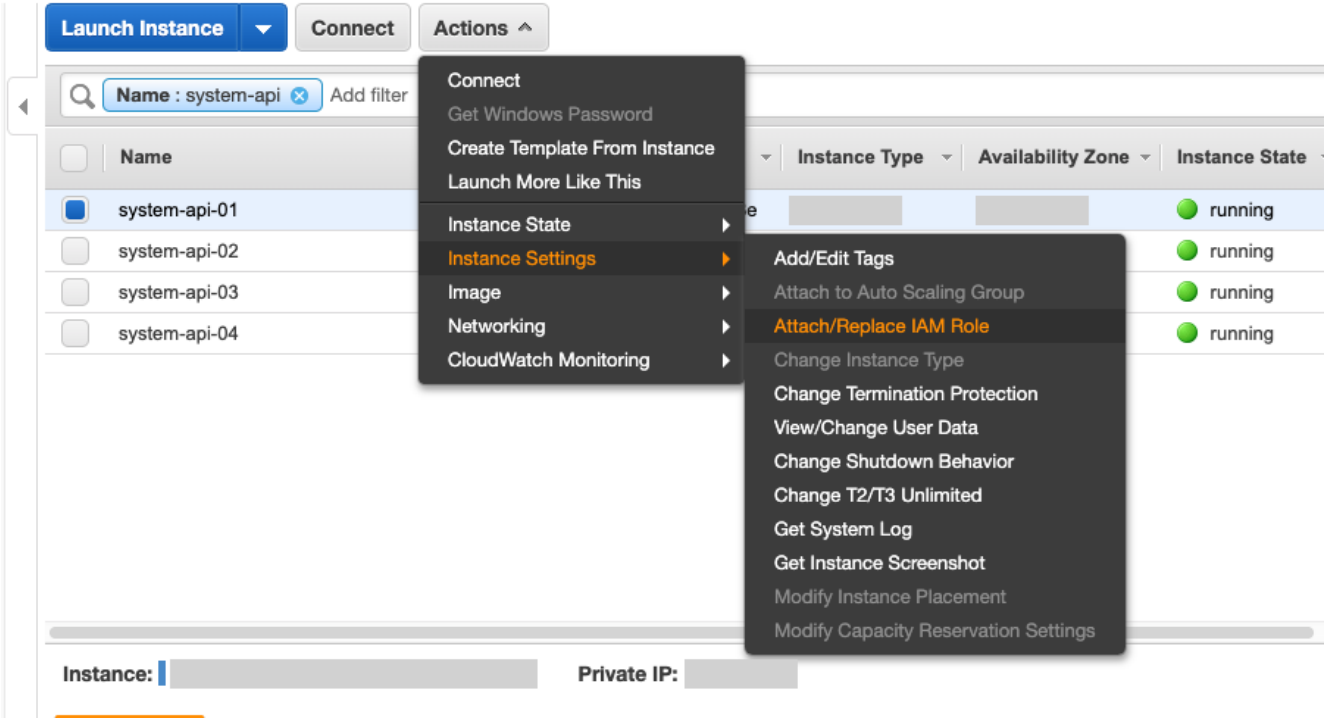
Create a New IAM Role Cont'd

- Search for the CloudWatchAgentServerPolicy policy, check the checkbox and click Next: Tags
- Add tags if required. Click Next: Review
- Enter a Role name (e.g. CloudWatchAgentServerRole). Then click Create role.



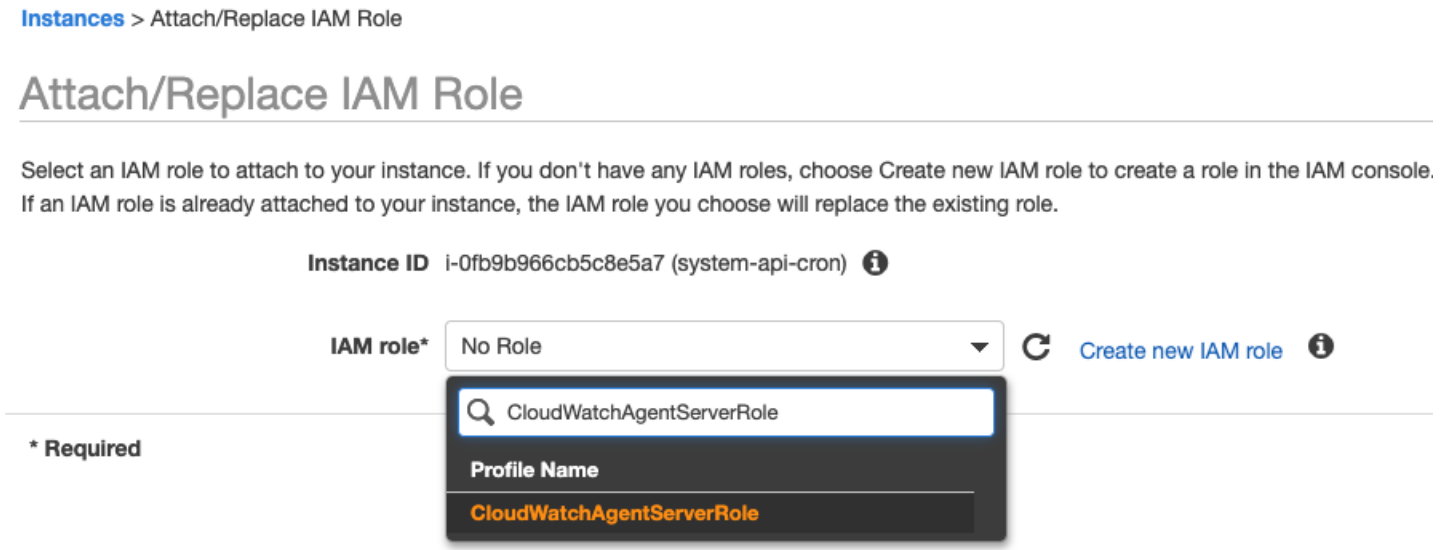
Attach the IAM Role

Go to the EC2 Dashboard, select **Instances** from the menu and check the checkbox next to the EC2 instance you want to stream the logs from. To attach the IAM Role, click the **Actions** dropdown and select **Instance Settings** > **Attach/Replace IAM Role:**



Attach the IAM Role Cont'd

Search for and select the IAM role created above (e.g. CloudWatchAgentServerRole), then click Apply to attach the IAM role:



Install CloudWatch Agent

- Connect to the EC2 instance running DV Server Platform
- First, download the CloudWatch Agent from S3
  - *wget https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb*
- Install the agent using the following command
  - *sudo dpkg -i -E ./amazon-cloudwatch-agent.deb*
- Now all we need to do is to configure the agent
  - Automatic using a wizard
  - Manually using a configuration file (we will be using this method since we are only interested in moving one or two files).

Manually Create config.json

The Log Agent uses a config file located at /opt/aws/amazon-cloudwatch-agent/bin/config.json.

Use your favourite editor (e.g. vim) to create and edit a file with the following content, e.g.

sudo vim /opt/aws/amazon-cloudwatch-agent/bin/config.json

```
{ "agent": { "run_as_user": "root"}, "logs": {"logs_collected": { "files": {"collect_list": [ {
                "file_path": "/opt/datavirtuality/dvserver/standalone/log/server.log",
                "log_group_name": "server-log",
                "log_stream_name": "{instance_id}"
}]}}}}
```

Manually Create config.json

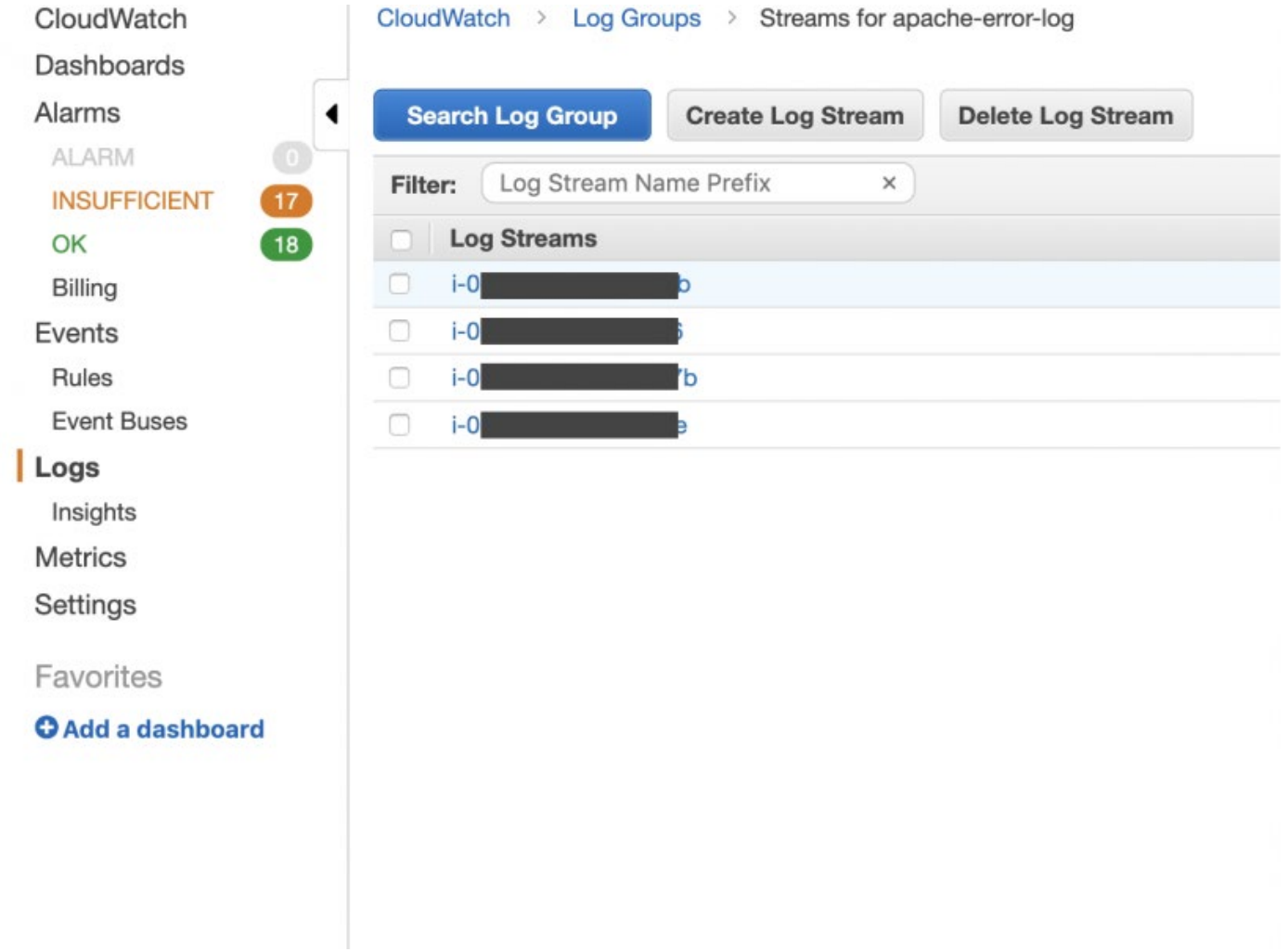
- The most important part of the config file is file_path.
  - This is the path to the log file on the server that you want to collect data from.
  - */opt/datavirtuality/dvserver/standalone/log/server.log* is the default log location for DV Platform.
  - The log_group_name and log_stream_name options are just used for naming the Log Group and Log Streams respectively in CloudWatch.
  - We recommend keeping {instance_id} for the log_stream_name as this helps identify which EC2 instance sent the log data.

Start the Agent

- Run the following command to run the agent. The CloudWatch agent is integrated with systemd so it will start automatically after a reboot:
  - *sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json -s*

View Logs

- Once the log file you are watching has data written to it, you'll be able to find it in **CloudWatch**. Go to the **CloudWatch Overview** and select **Logs** from the **menu**. You should see the label for the **Log Group** you used in the config (e.g. **server-log**).
- Click on the log group name to see the **log streams**. Each log stream uses the EC2 instance ID, so you know which EC2 instance logged the data:To search the logs, click the Search Log Group button. In the filter text box, enter a search term to search all your log files in one go.



Any feedback / questions?

Thank you!

Please feel free to contact us at:
presales@datavirtuality.com

or

visit us at:
datavirtuality.com