SnowWatch

Release 0.1

Jun 11, 2019

Contents:

SNOWWATCH

SNOWWATCH is a collection of SQL commands and automated data harvesting scripts designed to let you efficiently ingest data from multiple cloud environments into the Snowflake Cloud Data Warehouse.

The Snowflake SnowAlert project is used to provide policy enforcement and alerting to improve the security, compliance, and affordability of your cloud infrastructure.

Get right to business by *going to our Quickstart guide.*.

CHAPTER 1

What SNOWWATCH *is*

Working with many clients who implement cloud monitoring solutions (with various levels of success), SNOWWATCH is our best effort at a minimalist and reliable data ingestion pattern.

The pattern is meant to make cloud data ingestion repeatable and trustworthy. You are encouraged to take the basic patterns in this project and expand them with your business-specific needs. SNOWWATCH is simply a starting point with data engineering best-practices built in.

CHAPTER 2

What SNOWWATCH *is not*

SNOWWATCH is not a platform and it is not a magic solution. This project will not replace your security and site reliability teams.

However, when used responsibly, this collection of ingestion patterns could save you time and money by allowing your engineers to focus more on business logic and less on setup.

SNOWWATCH

2.1 Quickstart

To get started, you'll need to setup the following two components:

- SnowAlert from Snowflake for monitoring and alerting.
- Datasources for automated data ingestion from different clouds.

2.1.1 SnowAlert

To get started, see the *deployment documentation* to stand up Snowalert in your preferred environment.

2.1.2 Datasources

After Snowalert is ready, begin automated data ingestion from your desired sources by setting up one or more *data-sources*.

2.2 SnowAlert Deployment

A core part of the SnowWatch platform is the deployment of the SnowAlert security analytics framework.

This framework is provided as a docker image hosted at docker hub here There are 2 docker images:

1. The snowsec/snowalert which is responsible for generating the alerts and violations
2. The snowsec/snowalert-webui which is the administration interface for SnowAlert. This container hosts a web application that allows a user to input an alert/violation/supression query.

There are several ways to orchestrate the containers.

2.2.1 Kubernetes SnowAlert Deployment

Overview

Deploying SnowAlert in k8s is relatively straightforwared. It consists of a k8s Deployment type that is for the SnowAlert UI, and an associated service with the type LoadBalancer. Additionally there is a CronJob type that runs the SnowAlert runner. The secrets are all managed via k8s.

Note

Using the included manifests is a good start to get up and running, however you will want to adjust namespaces and create roles that are suitable for your environment.

Quickstart

To use the included manifests (warninig, they will deploy into the default namespace). Modify the sa-secrets.yaml file with base64 encoded values that are the result of the non-KMS install process found here (i.e. the values contained within the envs file, which is the result of the install process mentioned above).

To Deploy

Assuming you have kubectl configured and pointing to your cluster you can execute

```
$ kubectl apply -f manifests
```

from within the /infra/k8s/ directory.

2.2.2 Fargate SnowAlert Deployment

2.2.3 Linux SnowAlert Deployment

SNOWWATCH

2.3 Datasources

SNOWWATCH is designed to let you modularly select the datasources that are relevant for your needs. Each data source contains specific setup instructions.

The following datasources are currently ready for use:

- *AWS Datasource* -> this data source is used to monitor your AWS environment.

2.3.1 AWS Datasource

The SNOWWATCH AWS Datasource is designed to gather data about your AWS infrastructure in a private S3 bucket owned entirely by you.

We then setup ingestion pipelines from your S3 bucket into Snowflake for further analysis. The data in this S3 bucket is formatted in such a way that Snowflake can easily ingest the data as it arrives.

Serverless

This datasource leverages the Serverless Framework to deploy a series of Lambda functions and Cloudformation templates to start organizing your AWS data seamlessly.

None of this monitoring data will ever leave your environment (unless you grant external access to your S3 bucket, like what is done in the Snowflake Setup).

The following AWS resources are deployed by this datasource:

- S3 bucket for serverless deployments
- S3 bucket for storing your harvested data and cloudtrail
- Cloudtrail Trail
- 5 Lambda functions that gather data from AWS and saves the data as JSON files in S3
- IAM roles granting read access to your EC2, ELB, and IAM data

The data gathered by this serverless application is stored in your S3 bucket with the following directories:

- `cloudtrail` -> raw output from your Cloudtrail Trail (not easily consumed by Snowflake)
- `cloudtrail_monitoring` -> processed output from a Lambda function that monitors new cloudtrail logs (easily consumed by Snowflake)
- `ec2_monitoring` -> processed output from a Lambda function that monitors ec2 instance metadata
- `elb_monitoring` -> processed output from a Lambda function that monitors elastic load balancer metadata
- `iam_monitoring` -> processed output from a Lambda function that monitors users, groups, roles, and policies from IAM
- `security_group_monitoring` -> processed output from a Lambda function that monitors security group metadata

Snowflake

To enable realtime data ingestion from AWS, the following objects are created in Snowflake:

- an `AWS` schema in your `SNOWWATCH` database
- `LANDING_ZONE` tables in the `AWS` schema for holding ingested data
- an S3 external stage
- a json file format within the `AWS` schema for reading monitoring files generated by the AWS Lambda functions
- a series of snowpipes that auto-ingest json files from s3 as they arrive

Installation Instructions

See detailed installation instructions here.

AWS Datasource Installation

Serverless

Start by deploying the serverless components:

1. Clone the SNOWWATCH github repo
2. Open a terminal and `cd` to the `datasources/aws/serverless` directory
3. Make sure you have npm installed. This package manager is used to download the Serverless Framework CLI.
4. Run `npm i` to install the required dependencies into your current directory.
5. Run the following code to confirm everything is working smoothly. This will create deployable artifacts in the `.serverless` directory in the current directory.

```
./node_modules/serverless/bin/serverless package --account_id <your aws␣
↪account id> --aws_config_profile_name <the profile name in your ~/.aws/
↪config file containing the credentials you'd like to use for deployment>
```

   Make sure to put in your own values. For example:

```
./node_modules/serverless/bin/serverless package --account_id 12345 --aws_
↪config_profile_name adminProfile
```

   Examine the resulting cloudformation templates in `./.serverless` to ensure things look right before deploying.

6. Run the following code to deploy to AWS

```
./node_modules/serverless/bin/serverless deploy --account_id <your aws␣
↪account id> --aws_config_profile_name <the profile name in your ~/.aws/
↪config file containing the credentials you'd like to use for deployment>
```

   Make sure to put in your own values. For example:

```
./node_modules/serverless/bin/serverless deploy --account_id 12345 --aws_
↪config_profile_name adminProfile
```

At this point, the AWS datasource serverless application should be deploying to AWS Cloudformation. Your monitoring data can be viewed in the `snowwatch-<your aws account id>` S3 bucket.

NOTE: The AWS config profile that you use will need to have some level of AWS admin priveleges in order to deploy the serverless components.

This will deploy the necessary AWS components to your AWS

Snowflake

Log in to your Snowflake acccount and open a new worksheet.

If this is your first SNOWWATCH datasource, run the initial setup script here. This script will need to run by one of your Snowflake admins. Ensure that your admin grants usage of the `SNOWWATCH_ROLE` to you.

Next, you'll need to gather some text values before running the setup SQL for this datasource. You'll need:

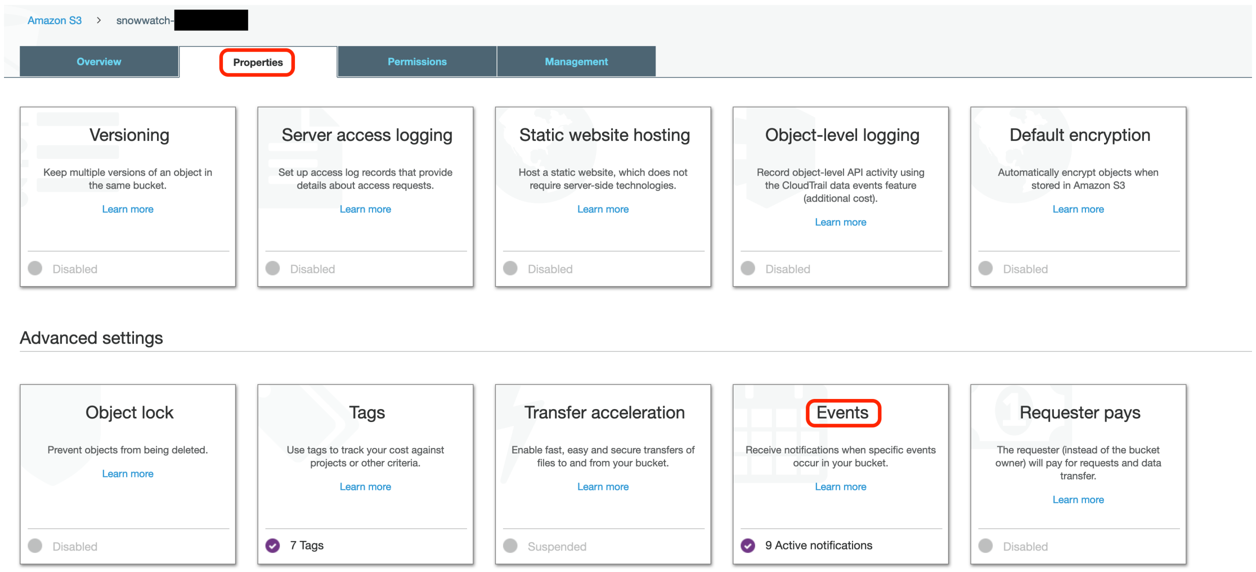
- the name of your serverless S3 bucket where monitoring reports are saved. This should be `snowwatch-<your aws account id>`. For example, `snowwatch-12345`.
- The AWS Key ID and Secret Key credentials you want Snowflake to use to access your S3 bucket. While there are other ways to grant Snowflake access to your AWS resources, this is the method Hashmap uses when building scripts. For information on how to set this up, and which IAM policies to assign to the owner of these credentials, see Snowflake's instructions here

Once you have these values, open a new worksheet in Snowflake and paste this AWS Setup SQL script into the worksheet. Place your S3 Bucket Name, AWS Key ID, and AWS Secret Key in the appropriate parts of the SQL script in the `CREATE STAGE ...` command.

Lastly, you'll need to create the `LANDING_ZONE` tables where your AWS monitoring data will be stored and setup your automated ingestion logic. To do this, run each SQL script here in its own Snowflake worksheet (do yourself a favor and name the worksheets to match the SQL script filenames).

The final command in each of the ingestion setup SQL scripts will display details about the current snowpipe. This information will include an AWS ARN to an SQS channel that the snowpipes are monitoring for new files. Unfortunately, to connect new data events to your snowpipes, you must manually add each snowpipe to your S3 bucket's notifications list. Luckily, each pipe has the same SQS ARN (because they share a stage).

Copy the ARN of the SQS queue that your snowpipes are monitoring and go to your `snowwatch-<your aws account id>` S3 bucket in the AWS web console. Go to the `Properties` Tab and select `Events`:



Versioning
Keep multiple versions of an object in the same bucket.
Learn more
Disabled

Server access logging
Set up access log records that provide details about access requests.
Learn more
Disabled

Static website hosting
Host a static website, which does not require server-side technologies.
Learn more
Disabled

Object-level logging
Record object-level API activity using the CloudTrail data events feature (additional cost).
Learn more
Disabled

Default encryption
Automatically encrypt objects when stored in Amazon S3
Learn more
Disabled

Advanced settings

Object lock
Prevent objects from being deleted.
Learn more
Disabled

Tags
Use tags to track your cost against projects or other criteria.
Learn more
✓ 7 Tags

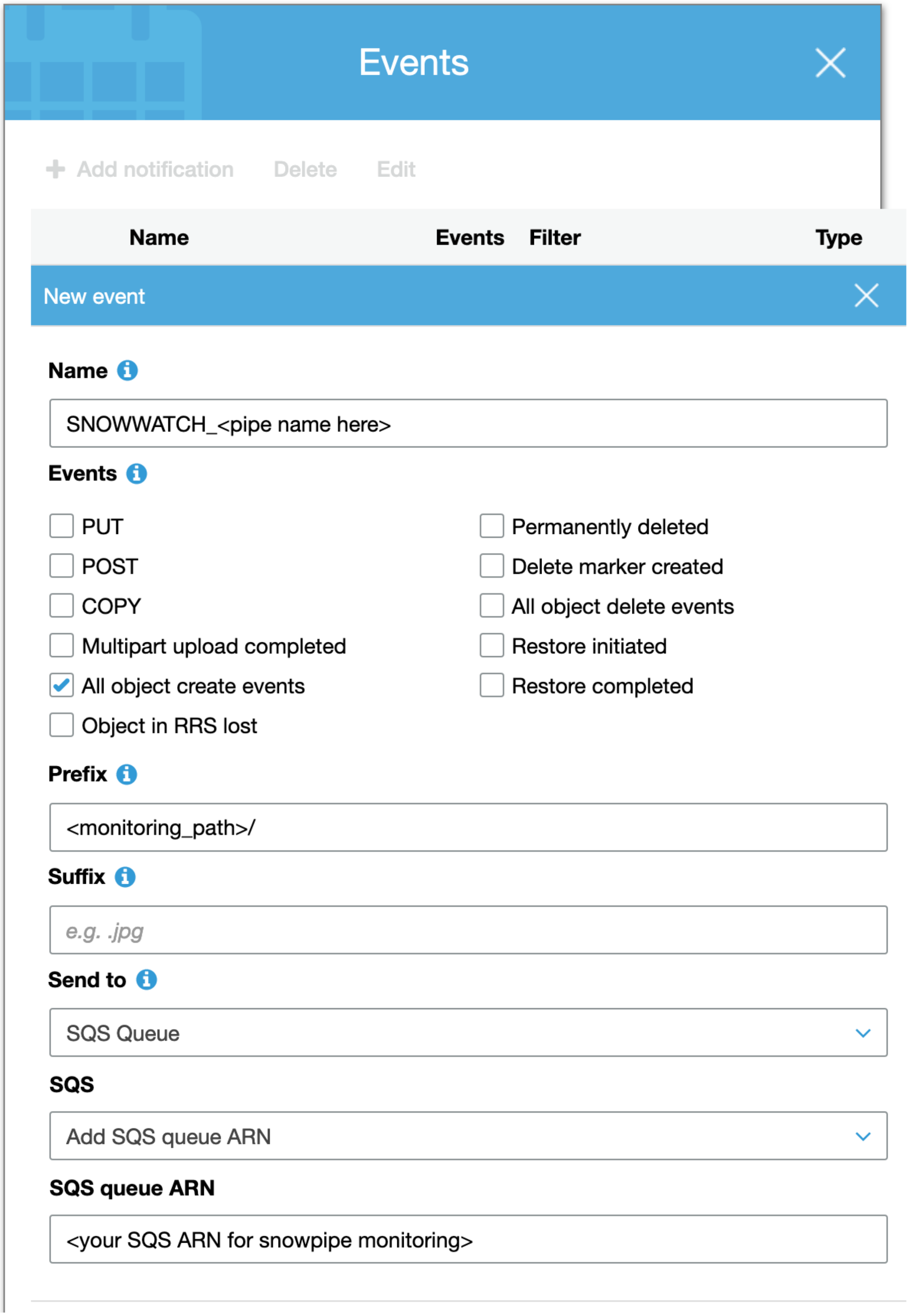
Transfer acceleration
Enable fast, easy and secure transfers of files to and from your bucket.
Learn more
Suspended

Events
Receive notifications when specific events occur in your bucket.
Learn more
✓ 9 Active notifications

Requester pays
The requester (instead of the bucket owner) will pay for requests and data transfer.
Learn more
Disabled

You should notice that there is already an event on your S3 bucket created by the Serverless deployment to react to new log creation from Cloudtrail.

Select `Add notification` and enter the following information:



Name

SNOWWATCH_<pipe name here>

Events

- [ ] PUT
- [ ] POST
- [ ] COPY
- [ ] Multipart upload completed
- [x] All object create events
- [ ] Object in RRS lost
- [ ] Permanently deleted
- [ ] Delete marker created
- [ ] All object delete events
- [ ] Restore initiated
- [ ] Restore completed

Prefix

<monitoring_path>/

Suffix

e.g. .jpg

Send to

SQS Queue

SQS

Add SQS queue ARN

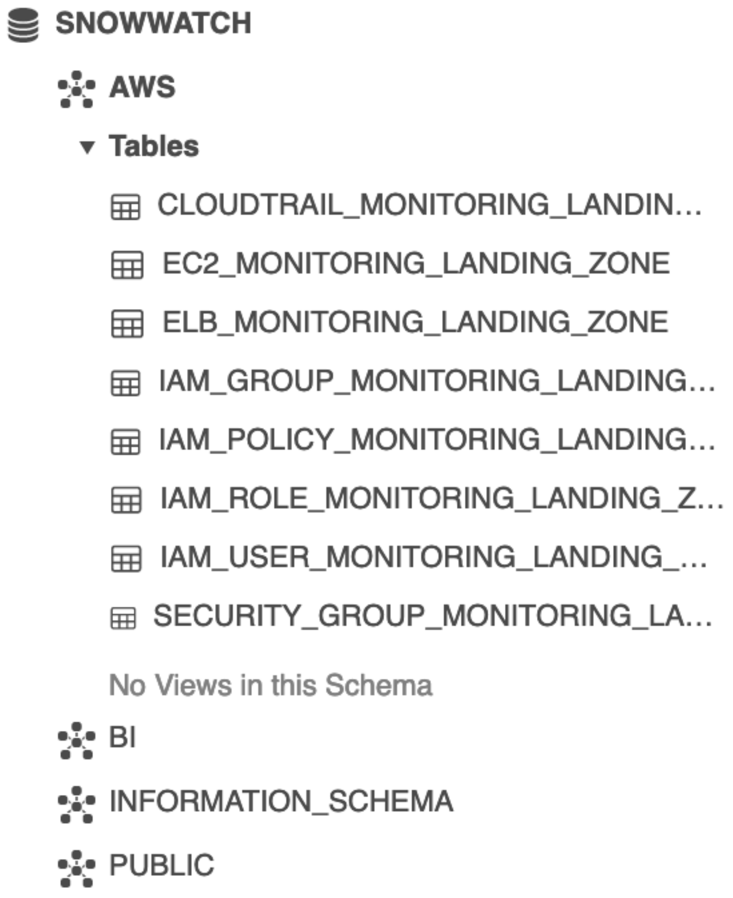
SQS queue ARN

<your SQS ARN for snowpipe monitoring>

You will have to create 8 total event notifications; 1 for each snowpipe. Each will share the same SQS ARN, but will have the following values:

Event Name	Prefix
SNOWWATCH_CLOUDTRAIL_MONITORING_PIPE	`cloudtrail_monitoring/`
SNOWWATCH_EC2_MONITORING_PIPE	`ec2_monitoring/`
SNOWWATCH_ELB_MONITORING_PIPE	`elb_monitoring/`
SNOWWATCH_SECURITY_GROUP_MONITORING_PIPE	`security_group_monitoring/`
SNOWWATCH_IAM_USER_MONITORING_PIPE	`iam_monitoring/users/`
SNOWWATCH_IAM_ROLE_MONITORING_PIPE	`iam_monitoring/roles/`
SNOWWATCH_IAM_GROUP_MONITORING_PIPE	`iam_monitoring/groups/`
SNOWWATCH_IAM_POLICY_MONITORING_PIPE	`iam_monitoring/policies/`

When you have finished, your `SNOWWATCH` database should look like the following:


SNOWWATCH
AWS
Tables
CLOUDTRAIL_MONITORING_LANDIN...
EC2_MONITORING_LANDING_ZONE
ELB_MONITORING_LANDING_ZONE
IAM_GROUP_MONITORING_LANDING...
IAM_POLICY_MONITORING_LANDING...
IAM_ROLE_MONITORING_LANDING_Z...
IAM_USER_MONITORING_LANDING_...
SECURITY_GROUP_MONITORING_LA...
No Views in this Schema
BI
INFORMATION_SCHEMA
PUBLIC

Data will now arrive automatically in your landing tables as new monitoring reports are generated in AWS.