Database Backup Overview

The data for our application is stored in a MySQL database. To backup this data, we periodically execute a MySQL dump and store the output in an AWS S3 Bucket. For geographic redundancy, we store these backups in the London AWS Region while our primary database runs on a VPS in New York. We chose to store backups with the following level of granularity:

- Hourly backups for the last 24 hours
- Daily backups for the last 7 days
- Weekly backups for the last month
- Monthly backups for the last year (on S3)
- Monthly backups older than 1 year (on Glacier)

Configuration

We control this backup through a series of Bash Scripts, Cron Jobs, and AWS S3 Lifecycle Rules. We have cron jobs configured on our server to (for each database) MySQL dump the database, zip the output, and transfer the output to AWS S3 over a secure connection. These cron jobs run at the frequency specified above for the corresponding backups. Once the backup files are on S3, we use Lifecycle Rules to delete backups after their expiration time has been reached and to transition monthly backups to their new storage class (glacier) after the specified time has been reached.

We notify of the status of the Backup system via email. We configured two email addresses for this purpose: sysadmin@kipswarehouse.com and backupstatus@kipswarehouse.com. We send a status email for each successfully completed backup from sysadmin@kipswarehouse.com to backupstatus@kipswarehouse.com with information about the backup. backupstatus@kipswarehouse.com is an alias which allows us to easily add and remove subscribers to the list who want to receive updates about the backup system status. For now, this address forwards to a junk email address dedicated to this use case so that a developer can log in and see the current status and history of the backup system without being inundated with hourly emails.

To configure this system from scratch one would need to schedule cron jobs that execute the `backup.sh` script included in the scripts folder of this repository to run with the specified frequencies. API keys and credentials have intentionally been removed. Configuring this script will require that you have an AWS Account with an available S3 bucket (preferably geographically isolated from your database deployment) and a mail server configured to send email (we use Migadu with Mailgun).

Restoring from Backup

To restore from an existing backup, download and unzip the backup you want to restore. Connect to your MySQL database through a SQL client such as Sequel Pro and create a new database for the project. Import the backup file you just downloaded. The database should be restored from backup.