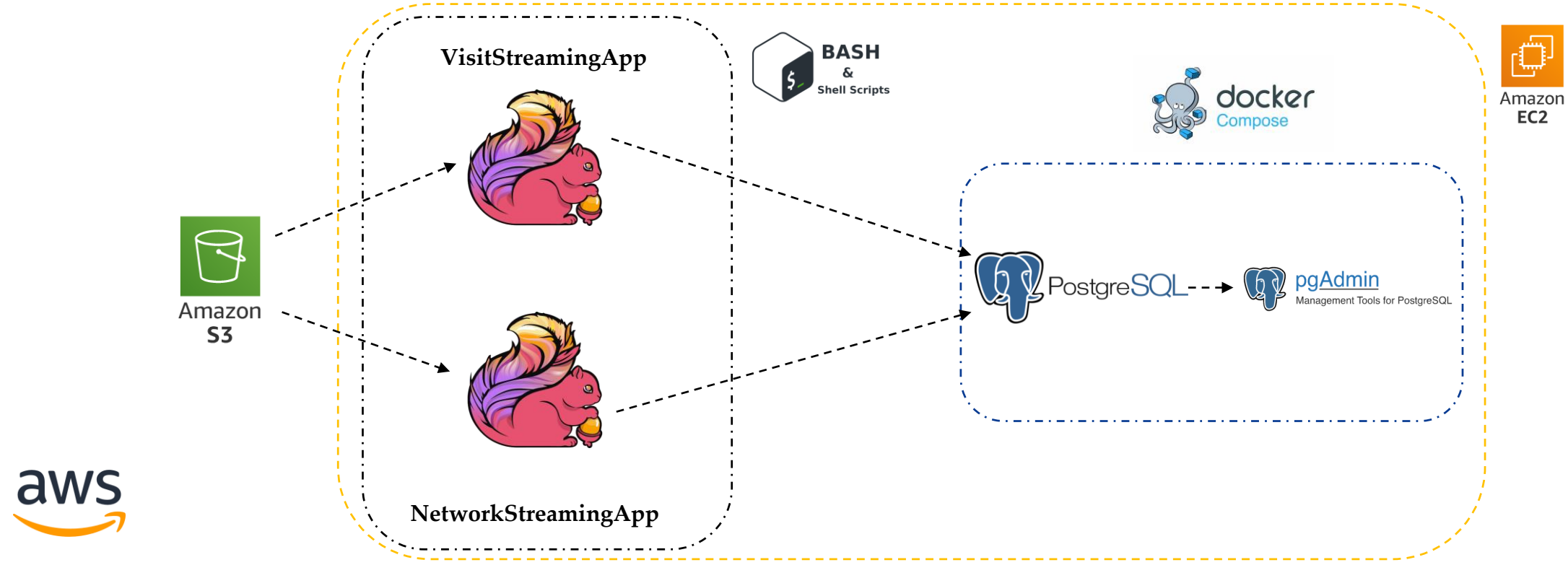
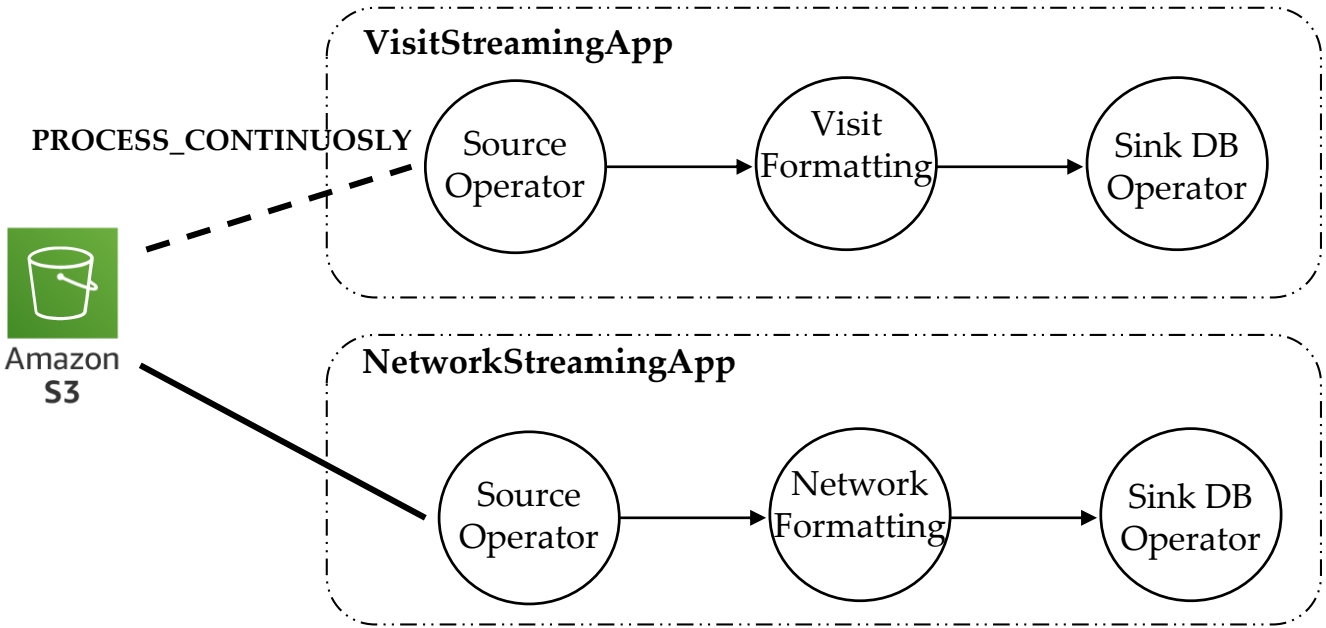
Satalia Data Engineering Challenge

Introduction

- **Objective:** Cloud-based Data Infrastructure, utilized by team of SQL Analysts
- **Initial Thought:** Design a solution using multiple AWS Services
  - Use a tech stack AWS Lambda, Kinesis or EMR and RDS along with Terraform for provisioning
  - Not too familiar with some of the technologies, not enough time to experiment
  - Limited by the free-tier options
- **Final Design:** Designed a fully custom infrastructure
  - Designed a custom infrastructure easily deployable on VMs
  - Interact only with S3 (raw data storage) and EC2 (VM to deploy infrastructure)
  - Still limited by free tier. Instance t3.micro offers 1GB


VisitStreamingApp
BASH
&
Shell Scripts
docker
Compose
Amazon
EC2
Amazon
S3
PostgreSQL
pgAdmin
Management Tools for PostgreSQL
NetworkStreamingApp
aws

Data Processing Applications



- Source Operators
  - **VisitStreamingApp:** *PROCESS_CONTINOUSLY* Mode to continuously fetch new files from S3
  - **NetworkStreamingApp**: Static Data fetch data once from S3
- Formatting Operators
  - **VisitStreamingApp:** Transform JSONL data and store in table
  - **NetworkStreamingApp:** Store Node Data in table
- Sink DB Operators
  - Sent batches of data to DB (1) when configured batch interval elapses, (2) max batch size is reached, (3) Flink Checkpoint has started

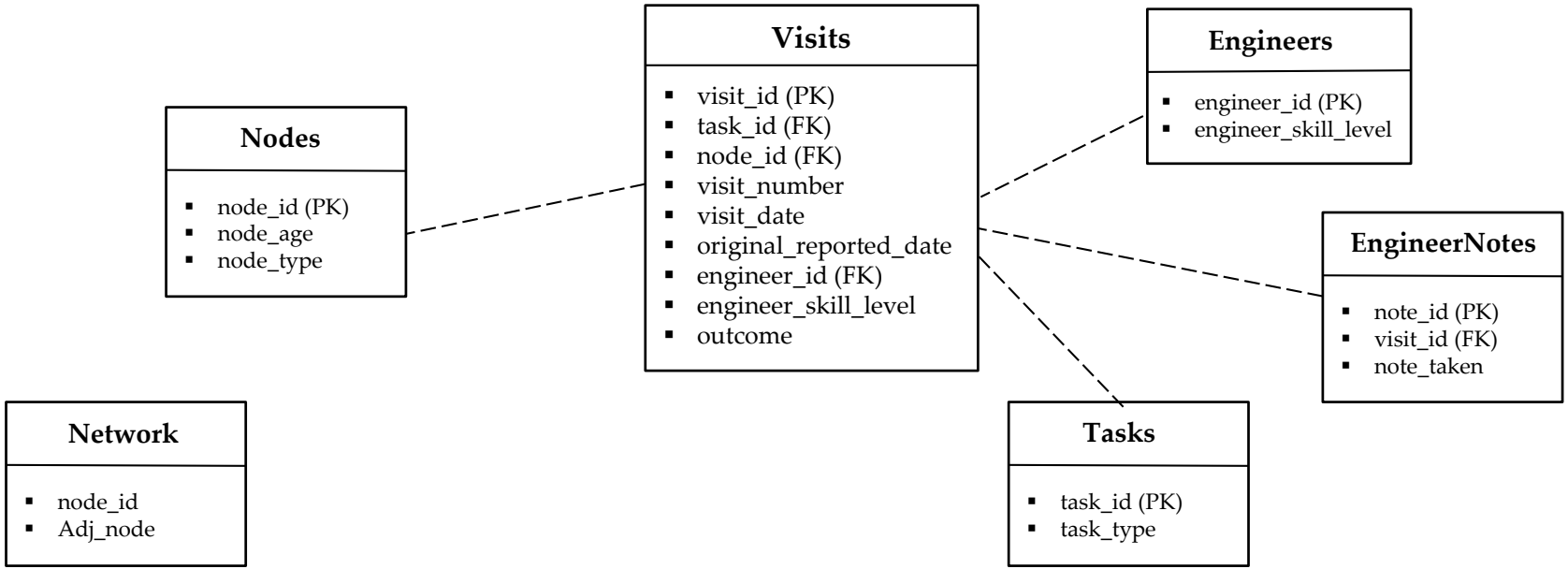
./manage.sh execution script

Execution Modes: **install | build | visit | network | stop | kill |**

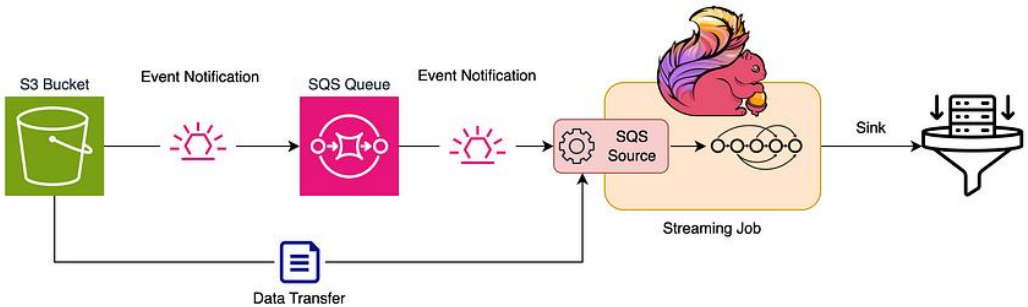
- **install**: Install the necessary software stack
  - Utilities (openjdk, maven, docker, docker-compose)
  - Apache Flink and necessary plugins
  - Apache Kafka (not used in the end)
- **build**: Build both VisitStreamingApp and NetworkStreamingApp
- **visit**: Executes VisitStreamingApp in Flink Cluster
- **network**: Executes NetworkStreamingApp in Flink Cluster
- **stop**: Stops Apache Flink Cluster and clean /tmp data
- **kill**: Terminate all services and clean all data (including docker service)

DEMO

Improvements #1: Data modelling



Improvements #2: Data fetching from S3



- Continuous polling in intervals
  (-) Too many requests
  (-) In case large interval, data can wait for long time
  (-) Default Source keeps track of already processed files in state store, cannot scale indefinitely

- Custom Source Operator
  Notification can trigger the processing of the newly uploaded file without waiting for a scheduled scan of the bucket
  (+) Reduced latency and overhead.

https://medium.com/datareply/event-driven-file-ingestion-using-flink-source-api-cfe45e43f88b