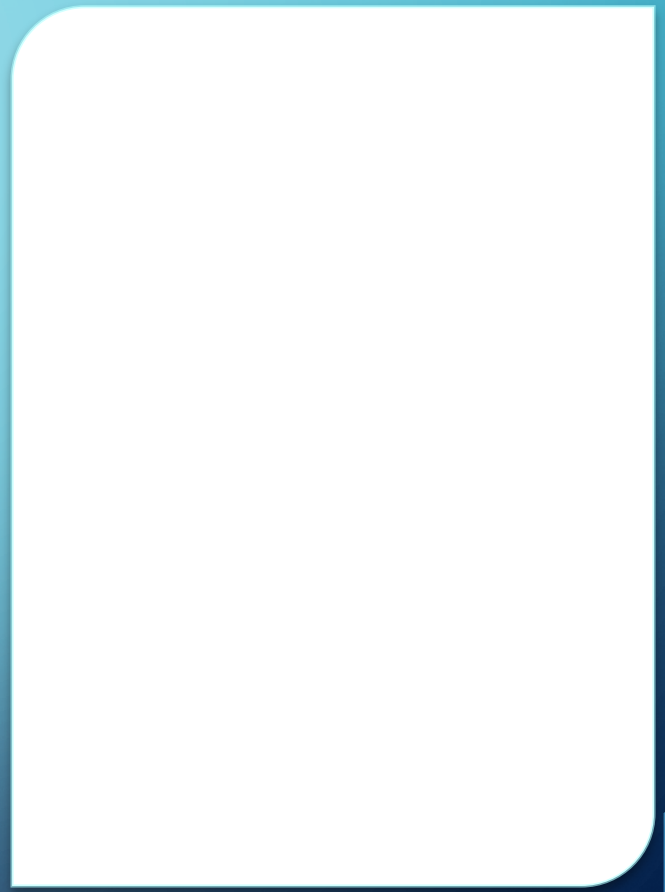

SELF HEALING BATCH JOBS
GLENN RENFRO
TWITTER @CPPWFS

GET IT OUT OF YOUR SYSTEM

BATCH CAN BE FUN

SCENARIO

SCENARIO



THREE STEPS TO BILLING

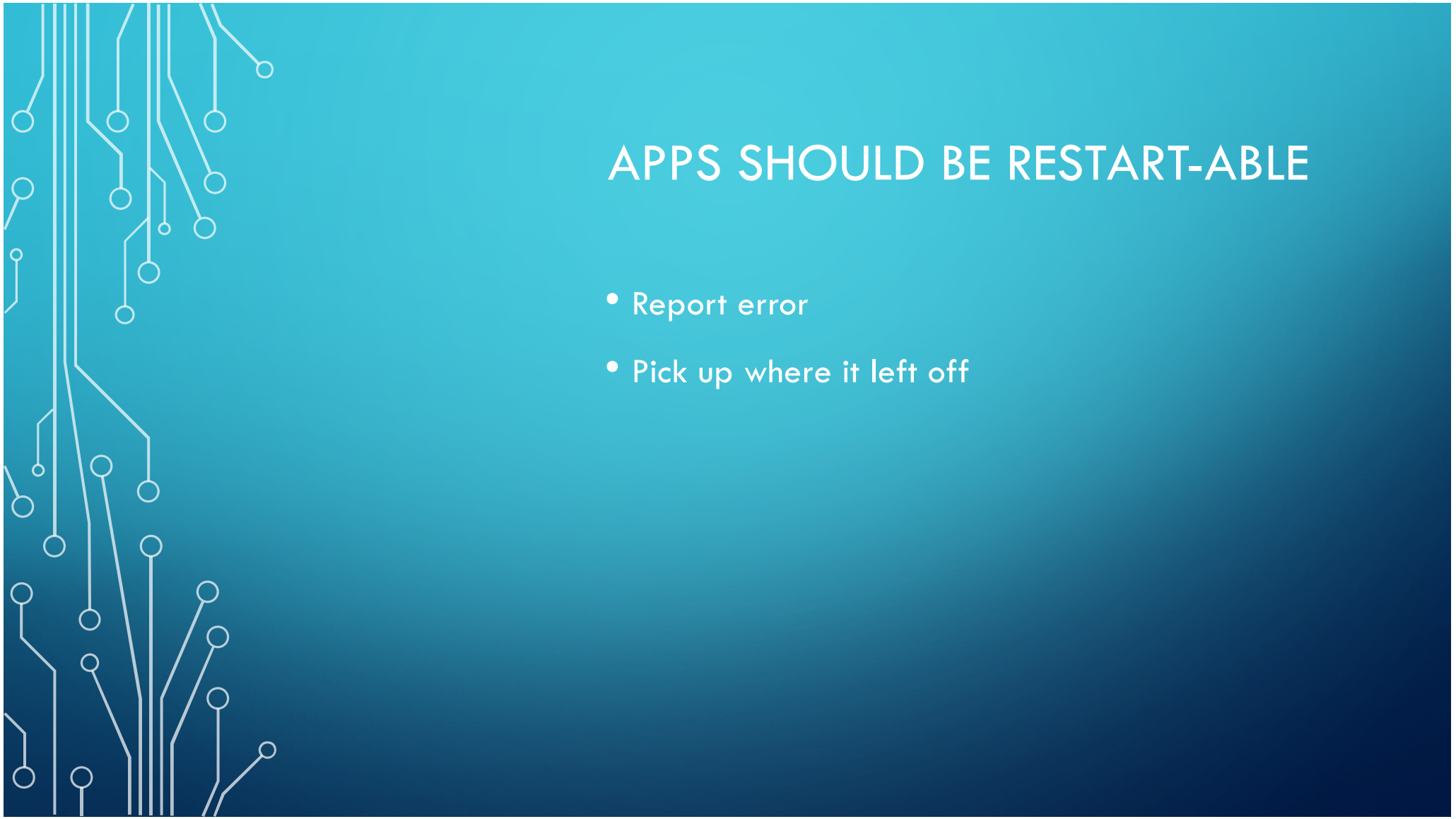
- Preparing Records
- Calculate and Record the bills
- Send out the bills

WHAT ACTION DO WE TAKE?

- App
  - Notification
  - Identify Cause
  - Pick up where we left off
- Infrastructure
  - Resolve
  - Relaunch

WHAT DO WE WANT TO DO?

- App should be restart-able
- Have system to capture and handle problem


APPS SHOULD BE RESTART-ABLE
• Report error
• Pick up where it left off


ISN'T THAT A LOT OF CODE?

Spring Batch Provides:

- Out of the box readers and writers
- Transaction support
- Restart-ability

Spring Cloud Task Provides:

- Report Status of an App to a DB
- Report Status of an App to Message Framework

```
@Bean
public Job fiveStep(BatchLabProperties batchLabProperties) {
  SimpleJobBuilder jobBuilder = jobBuilderFactory.get(batchLabProperties.getBatchName())
      .start(stepBuilderFactory.get(stepOneName)
          .tasklet(new Tasklet() {
            public RepeatStatus execute(...) throws Exception {
              logger.info("********ONE: Prepping Records********");
              return RepeatStatus.FINISHED;
            } }).build())
      .next(stepBuilderFactory.get(stepTwoName)
          .tasklet(new Tasklet() {
            public RepeatStatus execute(...) throws Exception {
              logger.info("********TWO: Calculating********");
              ...
            } }).build())
      .next(stepBuilderFactory.get(stepThreeName)
          .tasklet(new Tasklet() {
            public RepeatStatus execute(...) throws Exception {
              logger.info("********THREE: Writing Results********");
              return RepeatStatus.FINISHED;
            }
          }).build());
  return jobBuilder.build();
}
```

```
@Bean
public Job fiveStep(BatchLabProperties batchLabProperties) {
  SimpleJobBuilder jobBuilder = jobBuilderFactory.get(batchLabProperties.getBatchName())
      .start(stepBuilderFactory.get(stepOneName)
          .tasklet(new Tasklet() {
            public RepeatStatus execute(...) throws Exception {
              logger.info("********ONE: Prepping Records********");
              return RepeatStatus.FINISHED;
            } }).build())
      .next(stepBuilderFactory.get(stepTwoName)
          .tasklet(new Tasklet() {
            public RepeatStatus execute(...) throws Exception {
              logger.info("********TWO: Calculating********");
            ...
           } }).build())
      .next(stepBuilderFactory.get(stepThreeName)
          .tasklet(new Tasklet() {
            public RepeatStatus execute(...) throws Exception {
              logger.info("********THREE: Writing Results********");
              return RepeatStatus.FINISHED;
            }
          }).build());
  return jobBuilder.build();
}
```

```
@Bean
public Job fiveStep(BatchLabProperties batchLabProperties) {
  SimpleJobBuilder jobBuilder = jobBuilderFactory.get(batchLabProperties.getBatchName())
      .start(stepBuilderFactory.get(stepOneName)
          .tasklet(new Tasklet() {
            public RepeatStatus execute(...) throws Exception {
              logger.info("********ONE: Prepping Records********");
              return RepeatStatus.FINISHED;
            } }).build())
      .next(stepBuilderFactory.get(stepTwoName)
          .tasklet(new Tasklet() {
            public RepeatStatus execute(...) throws Exception {
              logger.info("********TWO: Calculating********");
              ...
            } }).build())
      .next(stepBuilderFactory.get(stepThreeName)
          .tasklet(new Tasklet() {
            public RepeatStatus execute(...) throws Exception {
              logger.info("********THREE: Writing Results********");
              return RepeatStatus.FINISHED;
            }
          }).build());
  return jobBuilder.build();
}
```

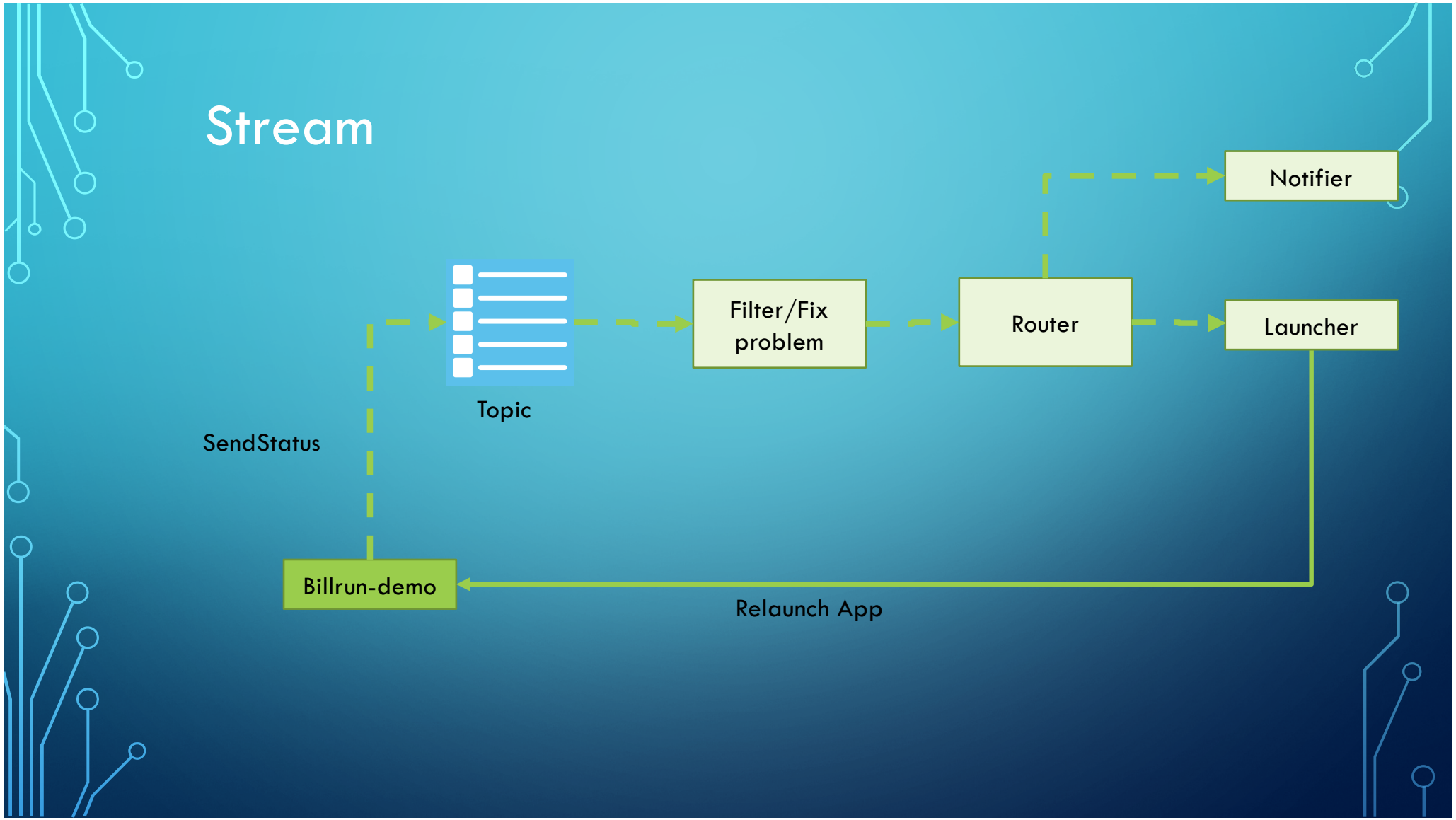
```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-task</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-stream-binder-rabbit</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-stream</artifactId>
</dependency>
```

```
@EnableBatchProcessing
@Configuration
@EnableTask
public class BatchDemoConfiguration {
...
}
```

RESTART-ABILITY ☐

FIX AND RESTART APP ☐

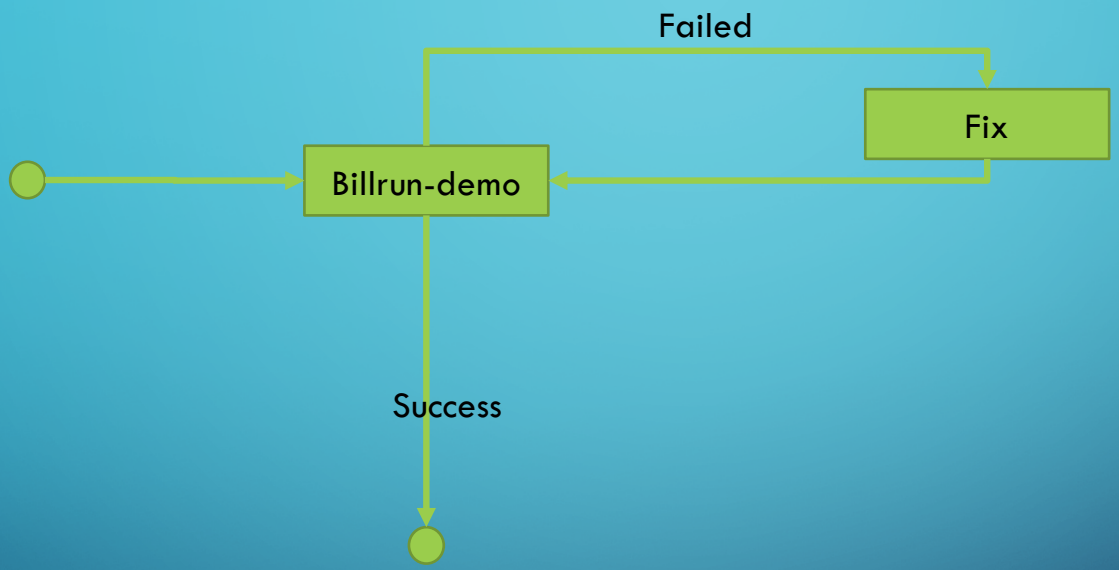
Stream


Notifier
Filter/Fix problem
Router
Launcher
Topic
SendStatus
Billrun-demo
Relaunch App

SPRING CLOUD STREAM
SPRING CLOUD STREAM APPS
SPRING CLOUD DATA FLOW

https://github.com/spring-cloud/spring-cloud-stream
https://github.com/spring-cloud/stream-applications
https://github.com/spring-cloud/spring-cloud-dataflow

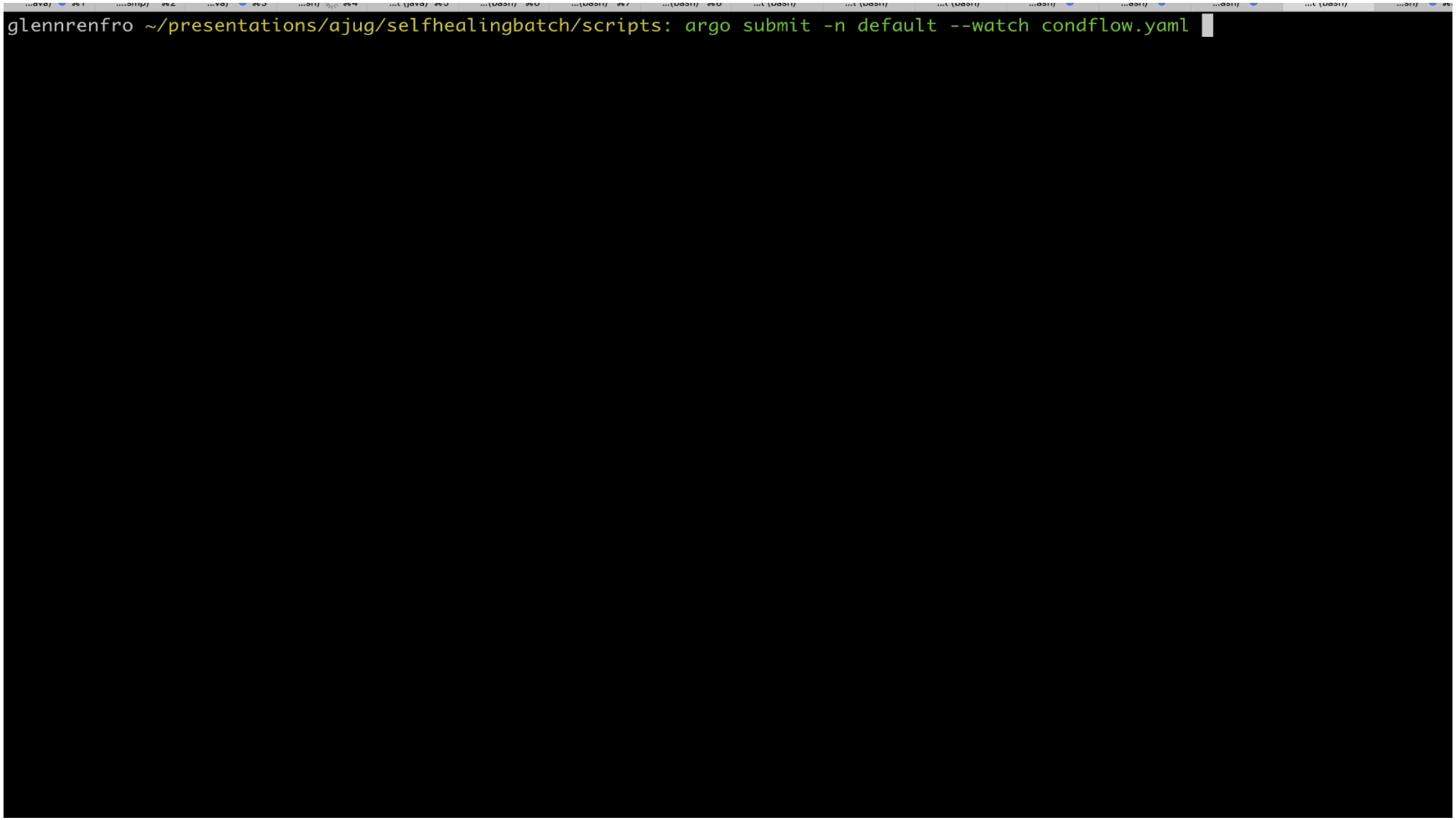
Workflow Engine



Workflow Engine

COMPONENTS

- Same billrunapp just dockerized using `./mvnw spring-boot:build-image`
- Workflowfix Boot application with Spring Cloud Task that resolves the problem
  - Also Dockerized using boot's build-image.


glennrenfro ~/presentations/ajug/selfhealingbatch/scripts: argo submit -n default --watch condflow.yaml



BUZZWORD BINGO


THANKS!