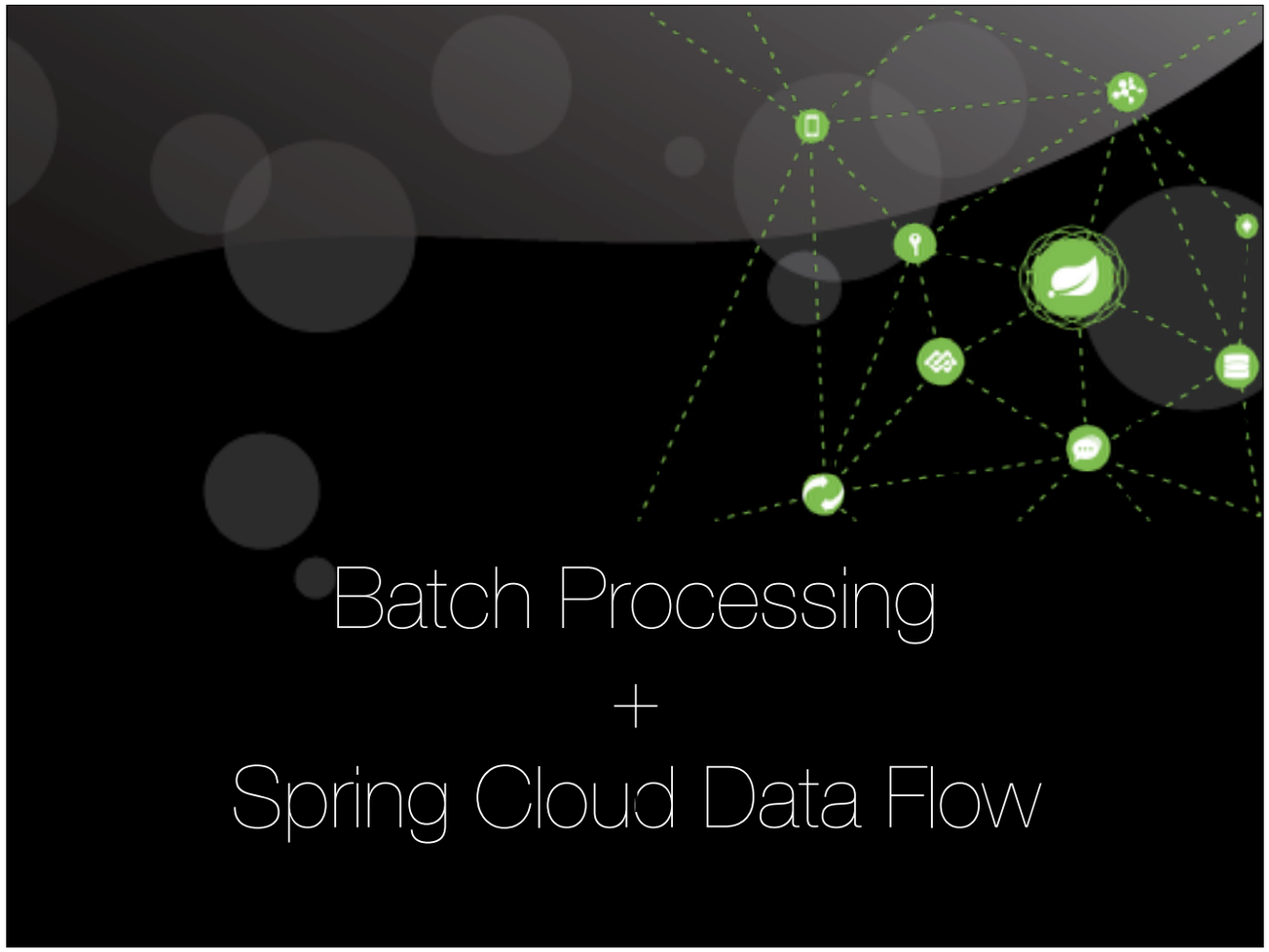
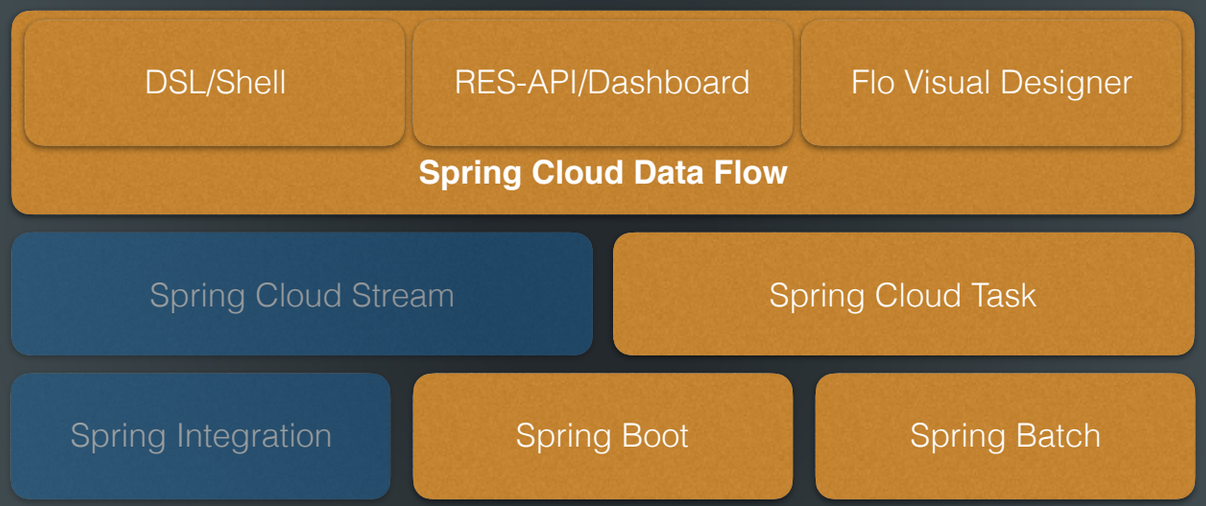

Batch Processing
+
Spring Cloud Data Flow


DSL/Shell
RES-API/Dashboard
Flo Visual Designer
Spring Cloud Data Flow
Spring Cloud Stream
Spring Cloud Task
Spring Integration
Spring Boot
Spring Batch

Batch Processing

Triggered or Scheduled	Mass Volumes of Data Processing	Remote Partitioning	Hybrid: Streaming and Batch
Short-lived	Transactional	Ordering	Offline Machine Learning
	Guarantees		

Before we get to far into what SCDF can do with tasks, let's review some Batch basics

- Some event kicks them off either automated (scheduled) or an event (user or some other Task or Long Running Process)
- Many times handles mass volumes of data
- Most of this work is transactional with some form of guarantee that it completes
- Batch processing may utilize Remote partitioning or ordering
- Example uses of batch apps
    - Batch Apps can be launched as a part of a stream
    - Or as offline machine learning projects.

Task = Short Lived

Spring Batch

enables the development of robust batch applications vital for the daily operations of enterprise systems

```
@EnableTask
```

Spring Cloud Task

enables you to develop and run short-lived executable data applications locally or in the cloud

Batch Processing in Spring Cloud Data Flow

Register you application

```
dataflow:>app register --name timestamp --type task --uri file:///tmp/
timestamp-task-1.1.0.RELEASE.jar
```


First thing you want to do is register your application
This notifies Spring Cloud DataFlow that this is an application is a task and can be used in task definitions.

Batch Processing in Spring Cloud Data Flow

Simple Task

```
dataflow:>task create foo --definition "timestamp"
```

Launch Task

```
dataflow:>task launch foo
```

Review Logs

- Once you have registered the application we want to create a task definition.
- A task definition is the establishes how a task execution should be executed.
- Now you can launch the task and check the results.
    - You can launch a task multiple times.
    - They can run simultaneously
- You can also put in properties for your task at
    - Task Definition time or

Batch Processing in Spring Cloud Data Flow

Set Properties at definition time

```
dataflow:>task create foo --definition "timestamp --format=YYYY"
```

Set Properties via arguments at launch time

```
dataflow:>task launch foo --arguments "--timestamp.format=YYYY"
```

Set Properties via properties at launch time

```
dataflow:>task launch foo --properties "app.timestamp.format=YYYY"
```


You can set properties via:

- Task Definition
- Arguments at launch time
- Properties at launch time

Getting the Info

Receive a list of available task definitions

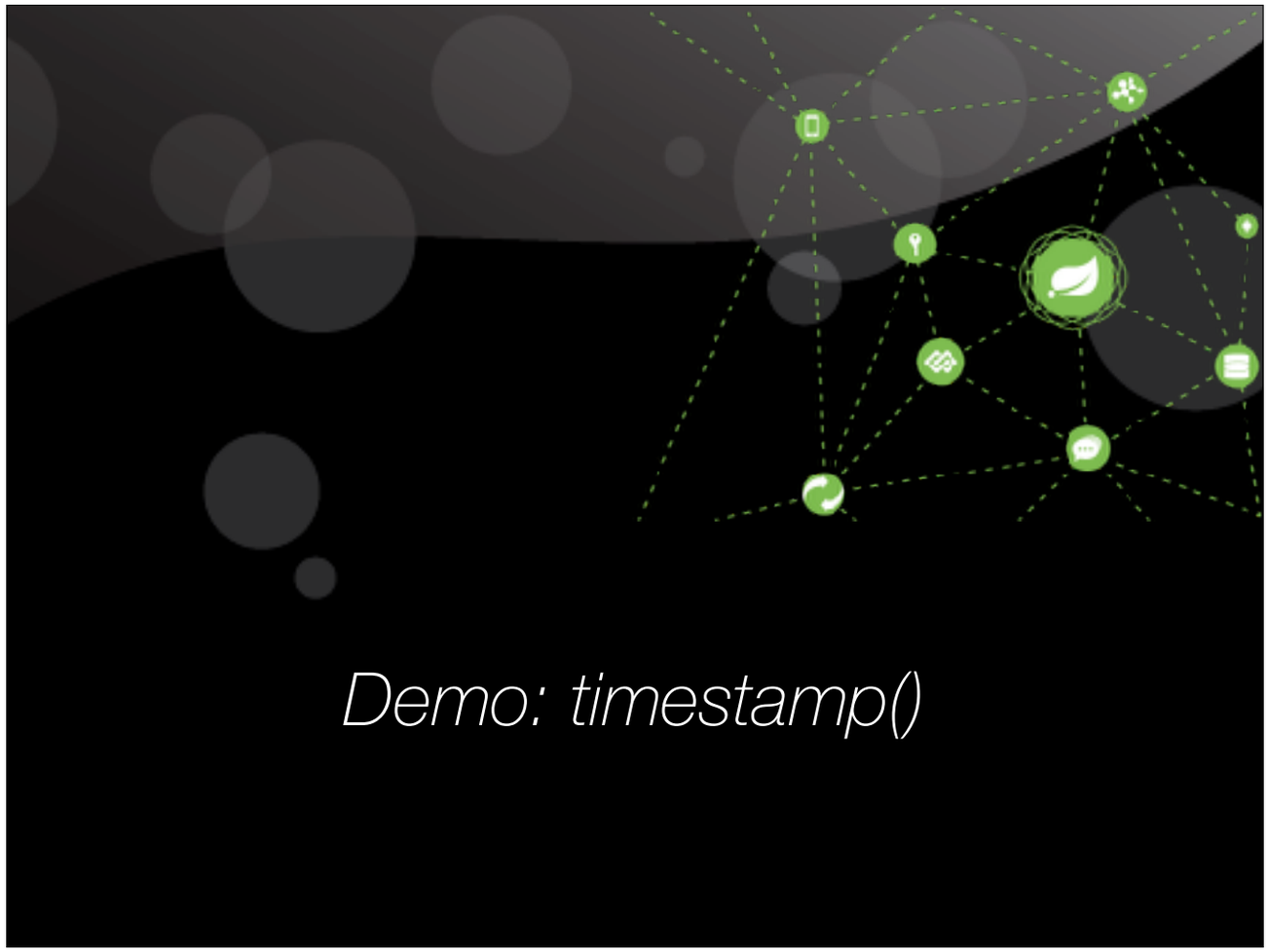
```
dataflow:>task list
```

Receive a list of previous task executions

```
dataflow:>task execution list
```

Get details on a specific task execution

```
dataflow:>task execution status --id 1
```


Demo: timestamp()

Task/Batch Lifecycle Events

Task events	*task-events*
Job Execution events	*job-execution-events*
Step Execution events	*step-execution-events*
Item Read events	*item-read-events*
Item Process events	*item-process-events*
Item Write events	*item-write-events*
Skip events	*skip-events*

When a user adds spring-cloud-stream-app-starters:

- Tasks will emit the following events to a stream:
  - task-events (Start Task, End Task)
- If the task is a Spring Batch the following events will be emitted:
  - job-execution-events
  - step-execution-events
  - item-read-events
  - item-process-events
  - item-write-events
  - skip-events

Lifecycle Events Subscription

Simple Task

```
dataflow:>task create foo --definition “batch-event”
```

Launch Task

```
dataflow:>task launch foo
```

TAP Task Events Channel

```
dataflow:>stream create bar --definition “:task-events > log” --deploy
```


In the example above we see that we have registered the batch-event example from Spring Cloud Task
https://github.com/spring-cloud/spring-cloud-task/tree/master/spring-cloud-task-samples/batch-events

From this we can receive just the task events as show above by creating stream that receives messages from a task-events destination.

Task Channel Destinations

Simple Task

```
dataflow:>task create foo --definition “batch-events”
```

Launch Task

```
dataflow:>task launch foo --properties “spring.cloud.stream.bindings.task-events.destination=myTaskEvents”
```

TAP Task Events Channel

```
dataflow:>stream create bar --definition “:myTaskEvents > log” --deploy
```


The example above is the same except we can direct the events to a specific destination by using the task-events.destination property.


Demo: Lifecycle Events

Stream and Batch Connectivity

Trigger a Task

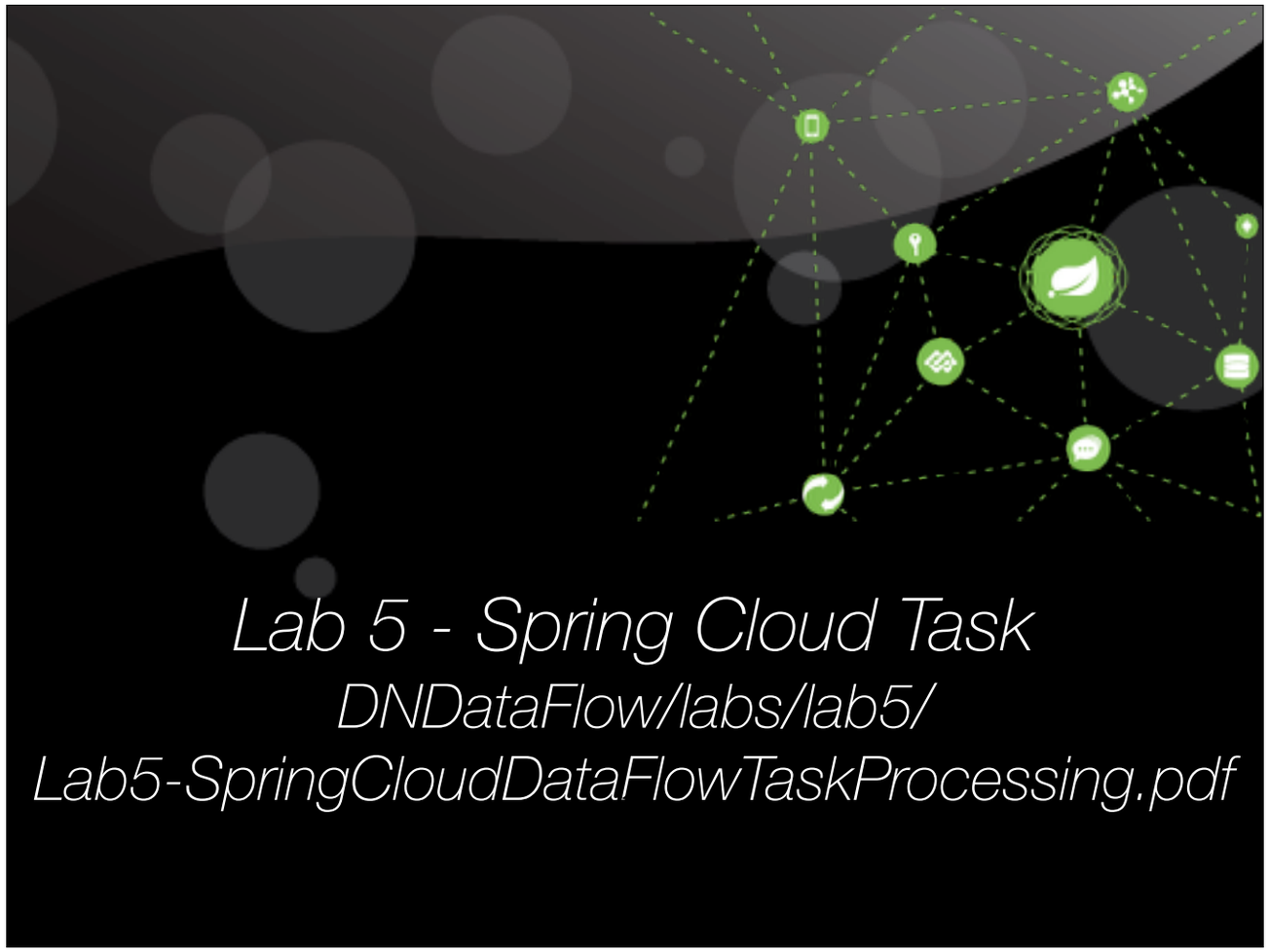
```
dataflow:>stream create foo --definition "triggertask --uri=maven://
org.springframework.cloud.task.app:timestamp-task:jar:1.0.0.BUILD-SNAPSHOT
--fixed-delay=5 | task-launcher-local"
```

Customize Task Launching

```
dataflow:>stream create bar --definition “http --port=9001 | my-task-
processor | task-launcher-local
```


In the example above we will launch the timestamp task once every 5 seconds using the trigger task.
Or create a my-task-processor that will transform a message to TaskLaunchRequest message and launch a specific . This is shown using the Spring Cloud Task sample found here: https://github.com/spring-cloud/spring-cloud-task/tree/master/spring-cloud-task-samples/taskprocessor


Demo: Stream and Batch


Lab 5 - Spring Cloud Task
DNDataFlow/labs/lab5/
Lab5-SpringCloudDataFlowTaskProcessing.pdf