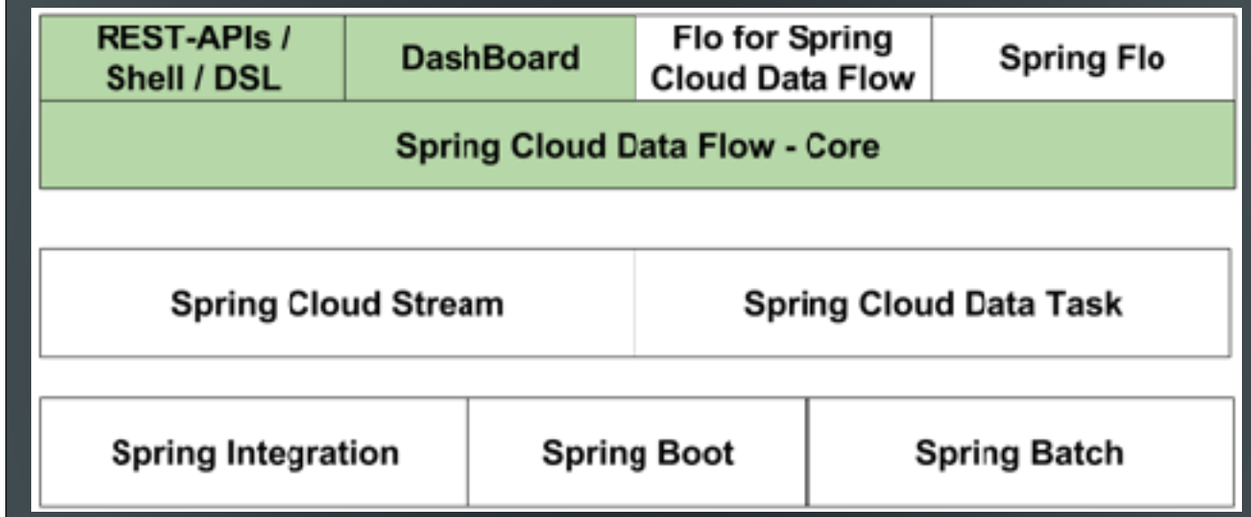
Spring Cloud Data Flow Architecture

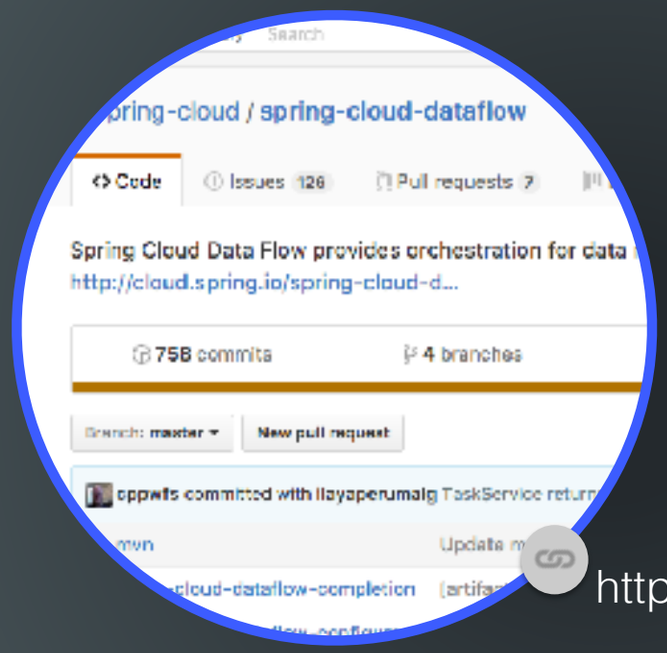
Glenn Renfro | Sabby Anandan

In this section we will cover
The major components of Spring Cloud Data Flow
Go into some detail of each and a quick overview of what they do.
Registering apps

The Big Picture



CODE IS ON GITHUB



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

JAR LINK

http://repo.spring.io/release/org/springframework/cloud/spring-cloud-dataflow-server-local/1.1.2.RELEASE/spring-cloud-dataflow-server-local-1.1.2.RELEASE.jar

https://flic.kr/p/9Bkxjo

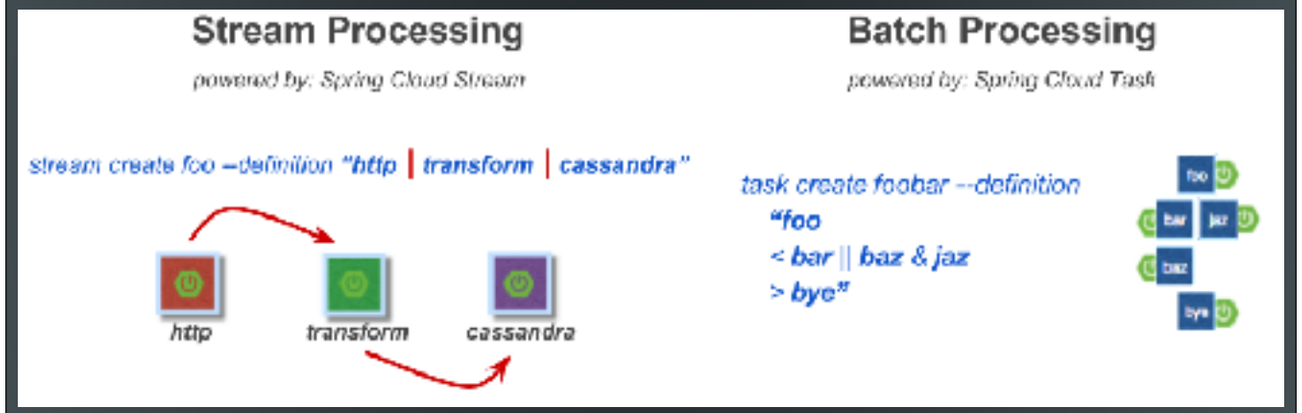
Reference Docs

http://docs.spring.io/spring-cloud-dataflow/docs/1.1.1.RELEASE/reference/htmlsingle/

https://flic.kr/p/8MVYfc

Microservices Architecture

Spring Cloud Data Flow is
a orchestration service for composable microservice applications on modern runtimes



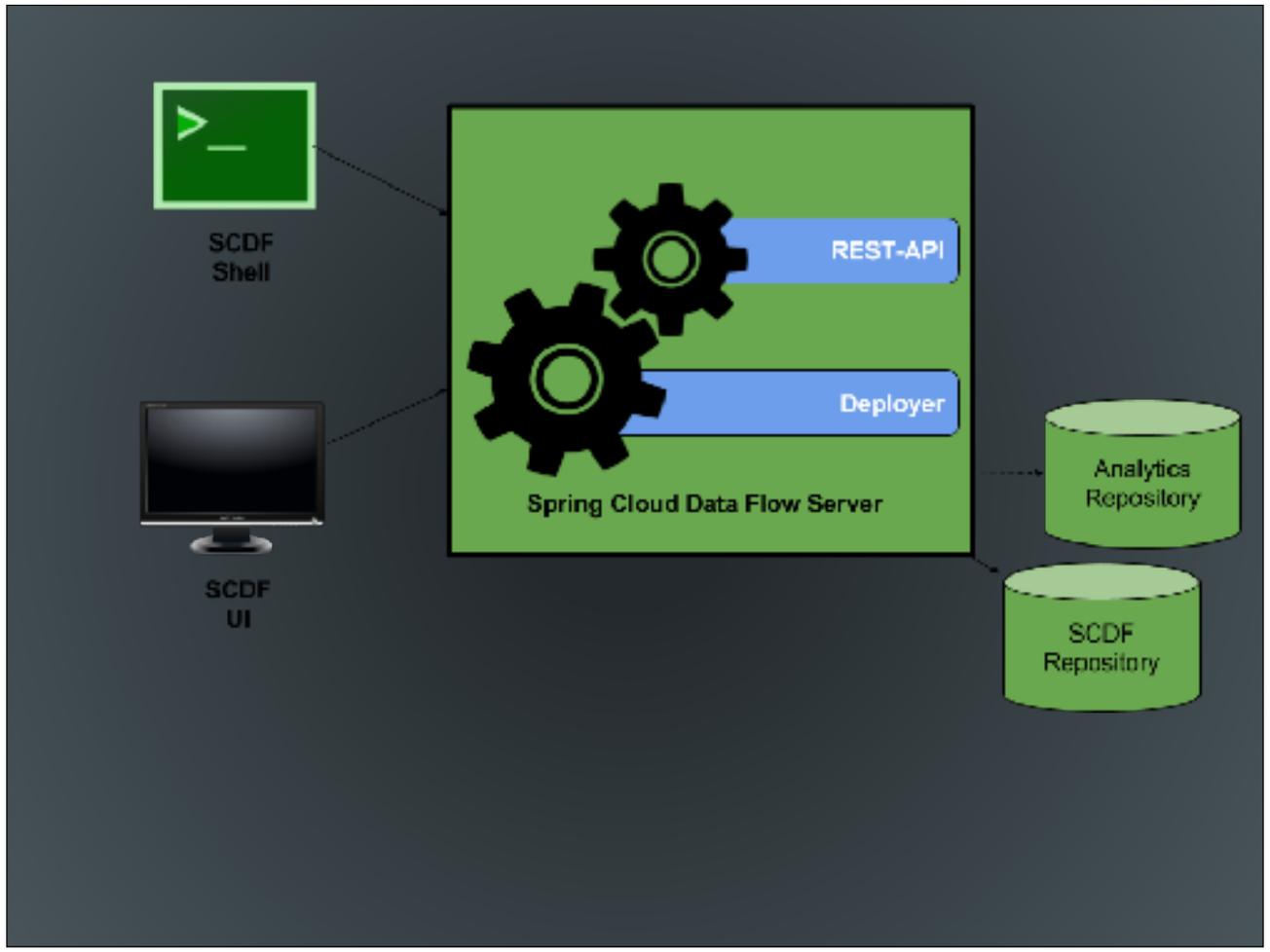
- So as we've discussed SCDF orchestrates the creation of composable microservice applications
- Streams and tasks are composed of applications that are deployed on the platform
-

Can Data Flow Help Me?

So how can Data Flow Help Me?

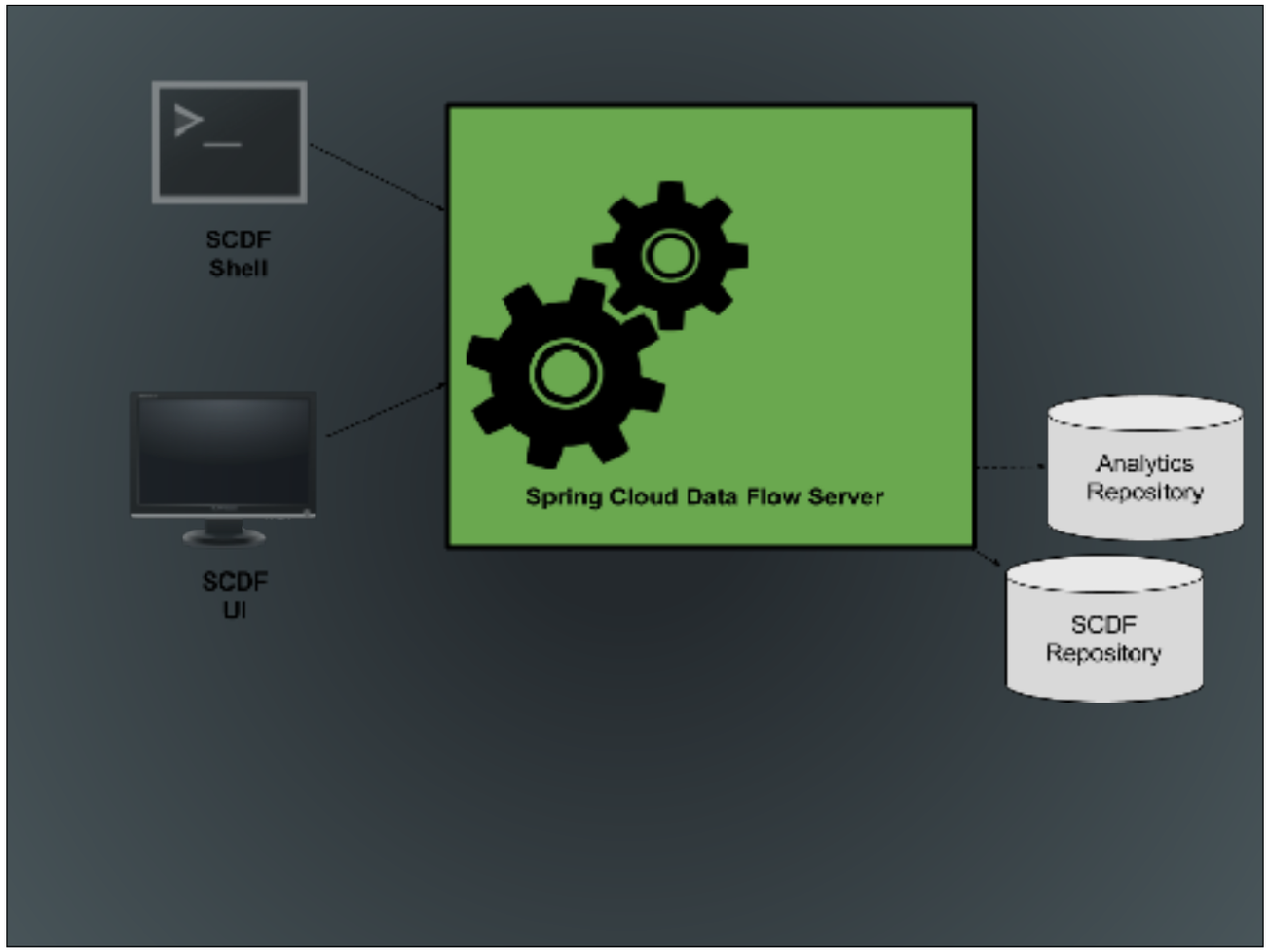
- Create Stream and task definitions easily via DSL, REST-APIs, Dashboard, and the drag-and-drop GUI - Flo
- Create, unit-test, troubleshoot and manage microservice applications in isolation
- Build data pipelines rapidly using the out-of-the-box stream and task/batch applications
- Consume microservice applications as maven or docker artifacts
- Scale data pipelines without interrupting data flows
- Orchestrate data-centric applications on a variety of modern runtime platforms including Cloud Foundry, Apache YARN, Apache Mesos, and Kubernetes
- Take advantage of metrics, health checks, and the remote management of each microservice application

https://flic.kr/p/4YzTmh

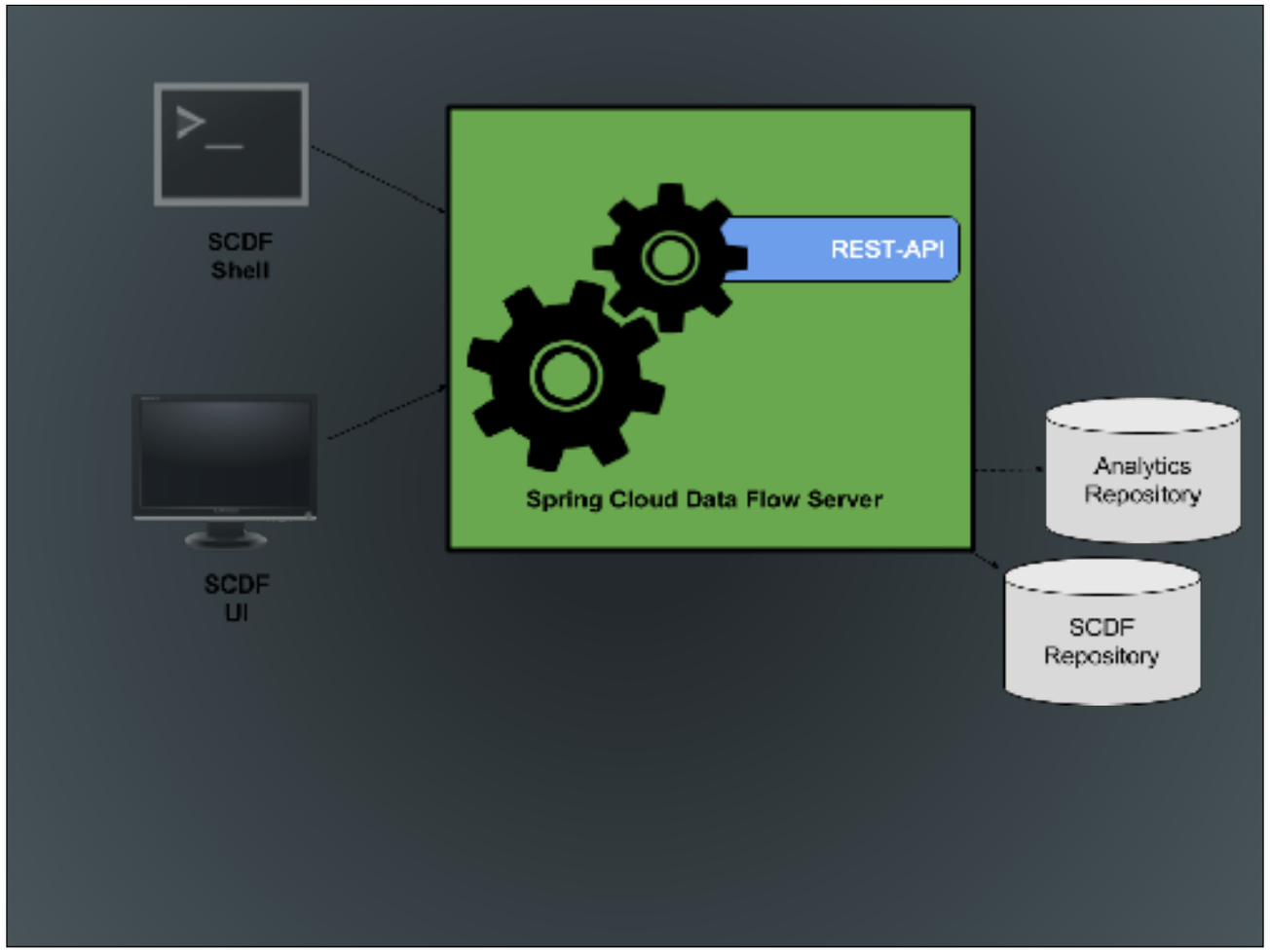


7 Components of the Spring Cloud Data Flow Server

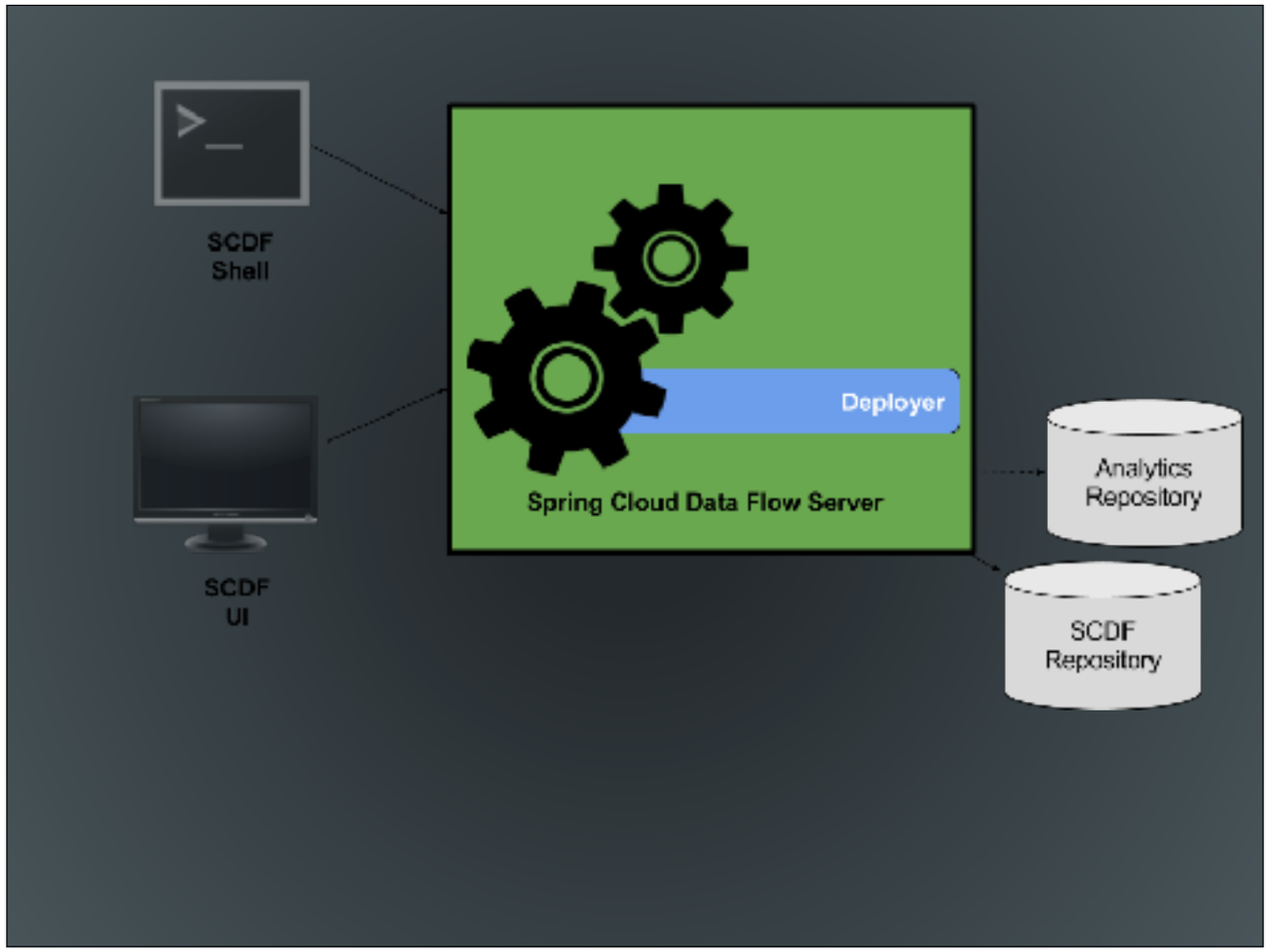
- Server -
- REST-API
- Deployer
- SCDF-UI
- SCDF Shell
- SCDF Repository
- Analytics Repository



- What is the Data Flow Server
  - The Data Flow Server uses an embedded servlet container and exposes REST endpoints for creating, deploying, undeploying, and destroying streams and tasks, querying runtime state, analytics, and the like.
- Allows us to deploy the applications that compose a stream. Or an launch application(s) for a task
- Via a restful-API or UI allows users to retrieve the state of the apps of a stream or task.
- Offers the ability to view the current values of the analytics
- Stores the the URI's of where to obtain the application.
- Stores the definitions for all the tasks and streams

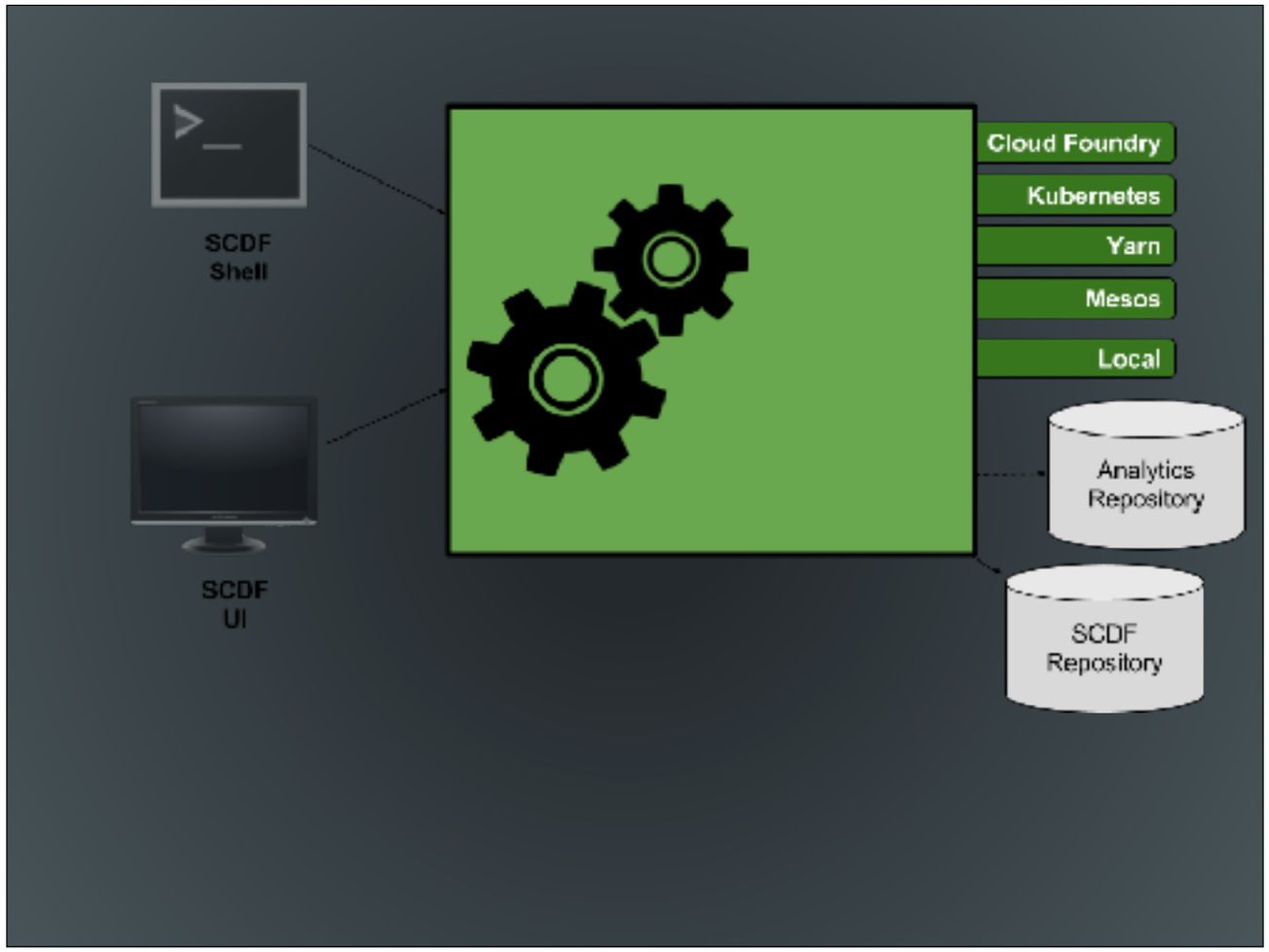


- Restful interface is offered from the SPI and thus all implementations support it.
- This is good for CI implementations
- Is build on Spring HATEOAS so it supports HATEOAS principles
  - Client doesn’t have to have prior knowledge of the server
- The endpoints are broken down into the following categories:
- Dashboard UI



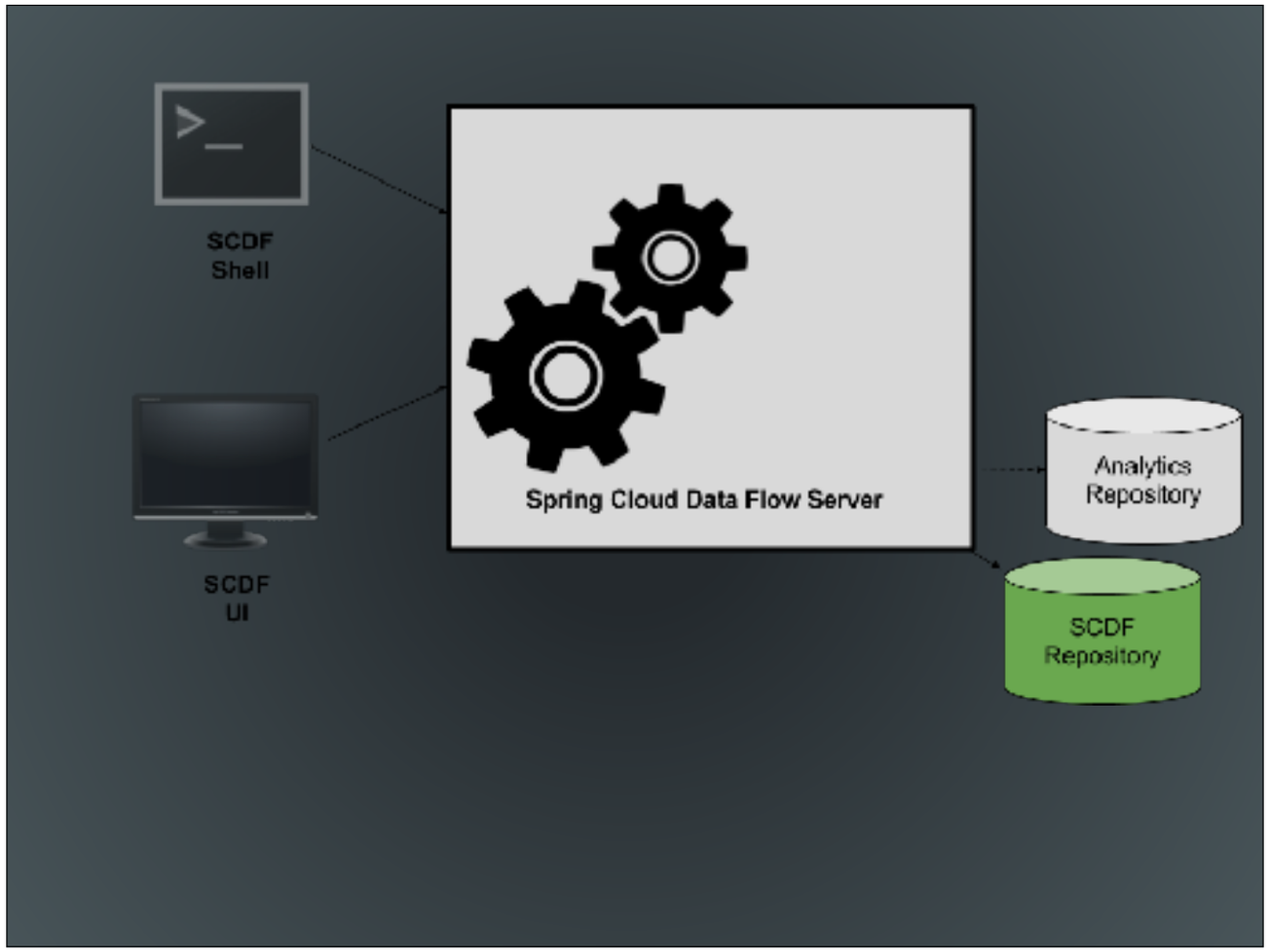
Spring Cloud Deployer

- Provides a common means to deploy applications to a platform
- Based on the Spring Cloud Deployer project https://github.com/spring-cloud/spring-cloud-deployer
- Each Spring Cloud Data Flow Server implementation uses one deployer. The current deployers that we support are:
  - CF
  - Mesos
  - Kubernetes
  - Yarn
  - Local
  - Others have been added by the community



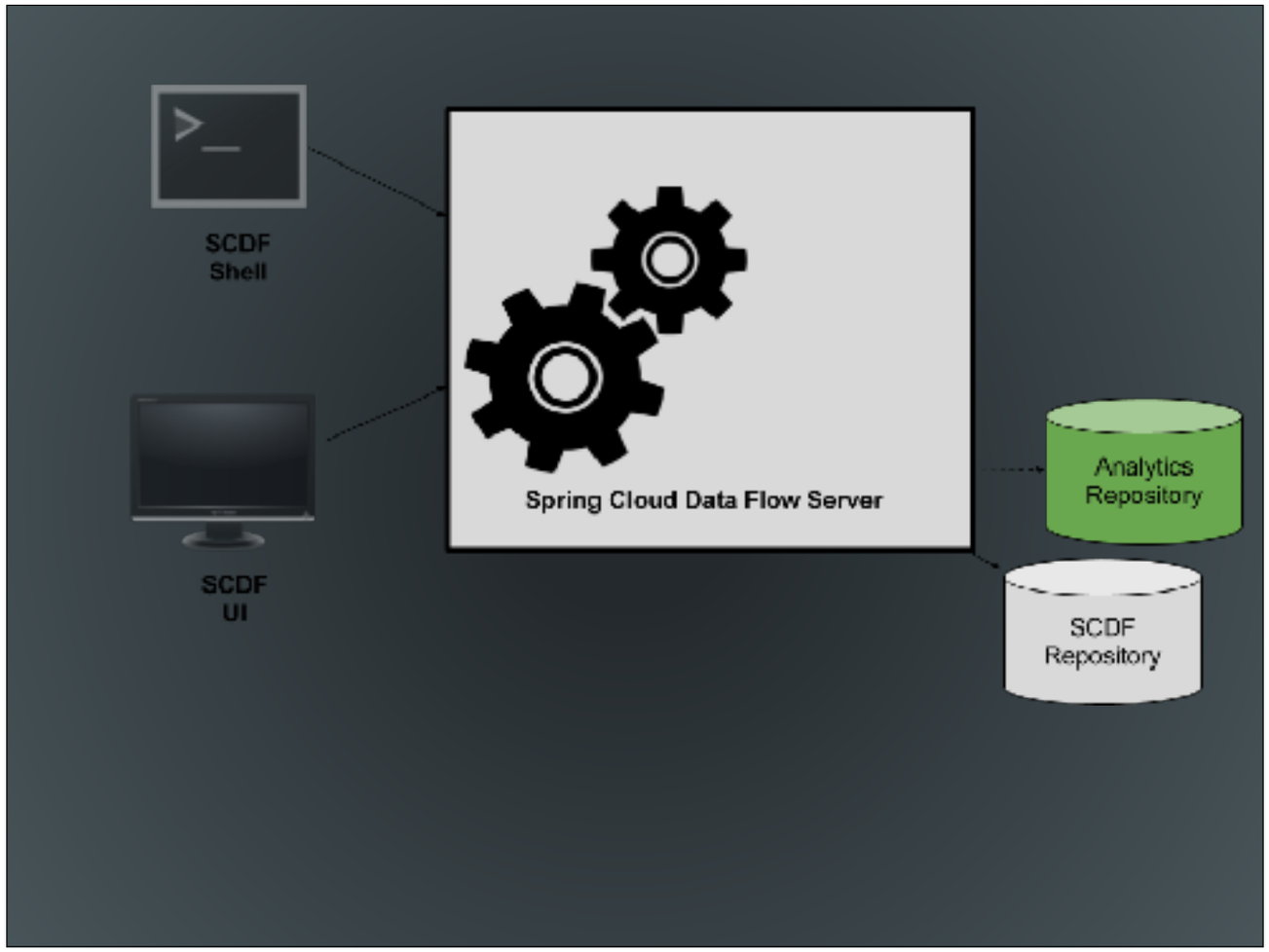
SCDF Server Types

- Spring Cloud Dataflow has a separate server for each deployment type.
- Cloud Foundry - https://github.com/spring-cloud/spring-cloud-dataflow-server-cloudfoundry
- Mesos - https://github.com/spring-cloud/spring-cloud-dataflow-server-mesos
- Yarn - https://github.com/spring-cloud/spring-cloud-dataflow-server-yarn
- Kubernetes - https://github.com/spring-cloud/spring-cloud-dataflow-server-kubernetes
- Local -SPI
- But all are based on the Spring Cloud Data Flow SPI - https://github.com/spring-cloud/spring-cloud-dataflow
- Can you support multiple platforms on a single SCDF instance.
- The answer is no. Each server supports only one platform.
- Why we are using Local
    - Meant for development purposes
    - Wanted to run it locally on your Machines
    - Wanted fast deployment to speed up labs
    - Wanted simple install (Yarn isn’t easy to install)



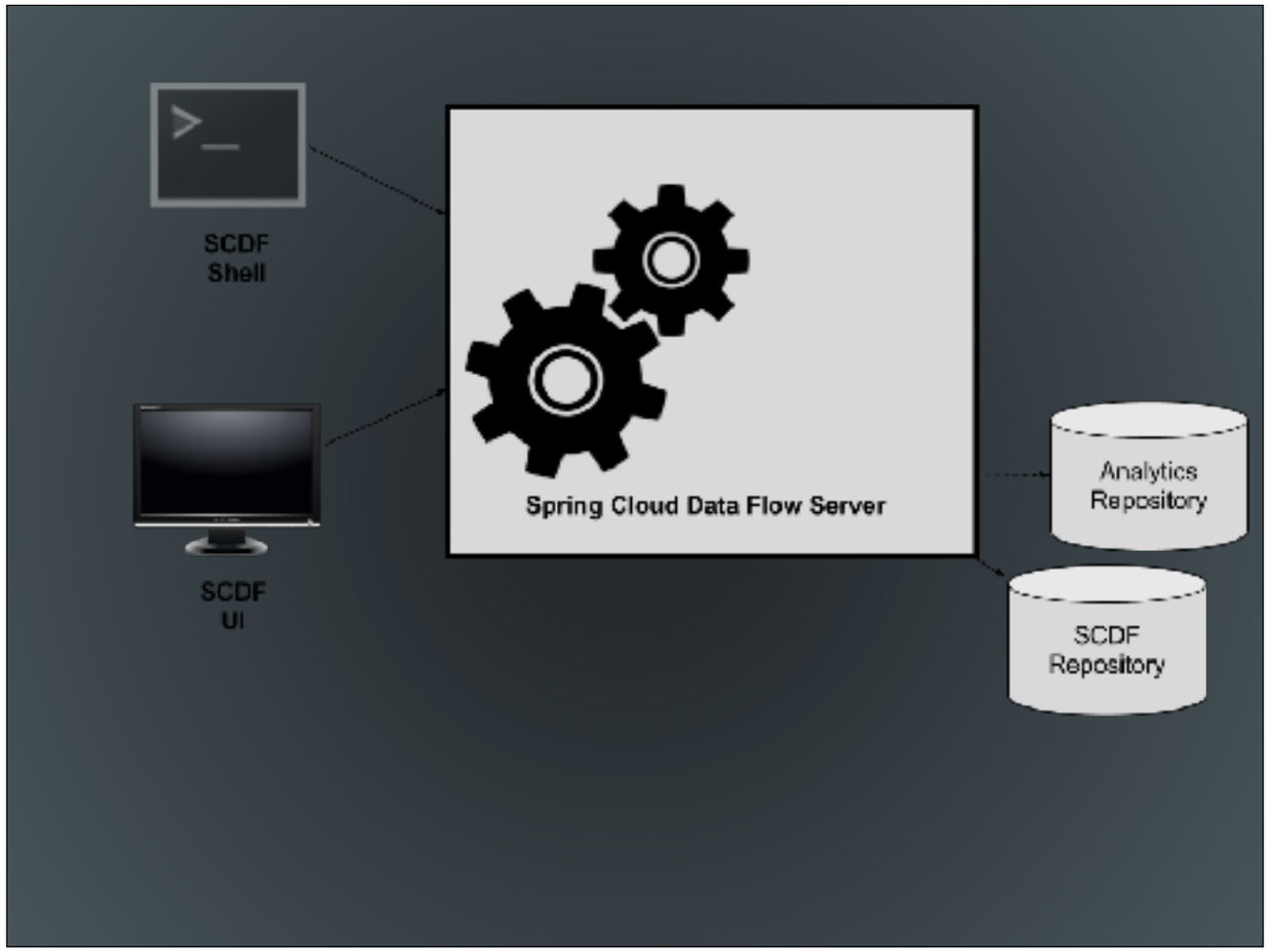
SCDF Repository

- Is an external relational database that stores:
  - Stream Definitions
  - Task Definitions
  - URI’s to the apps
  - Task Execution Statuses
  - Job Statuses
- By default Local uses an embedded H2 database
- Currently supported (out of the box) H2, HSQLDB, MySQL, POSTGRESQL
-



Analytics Repository

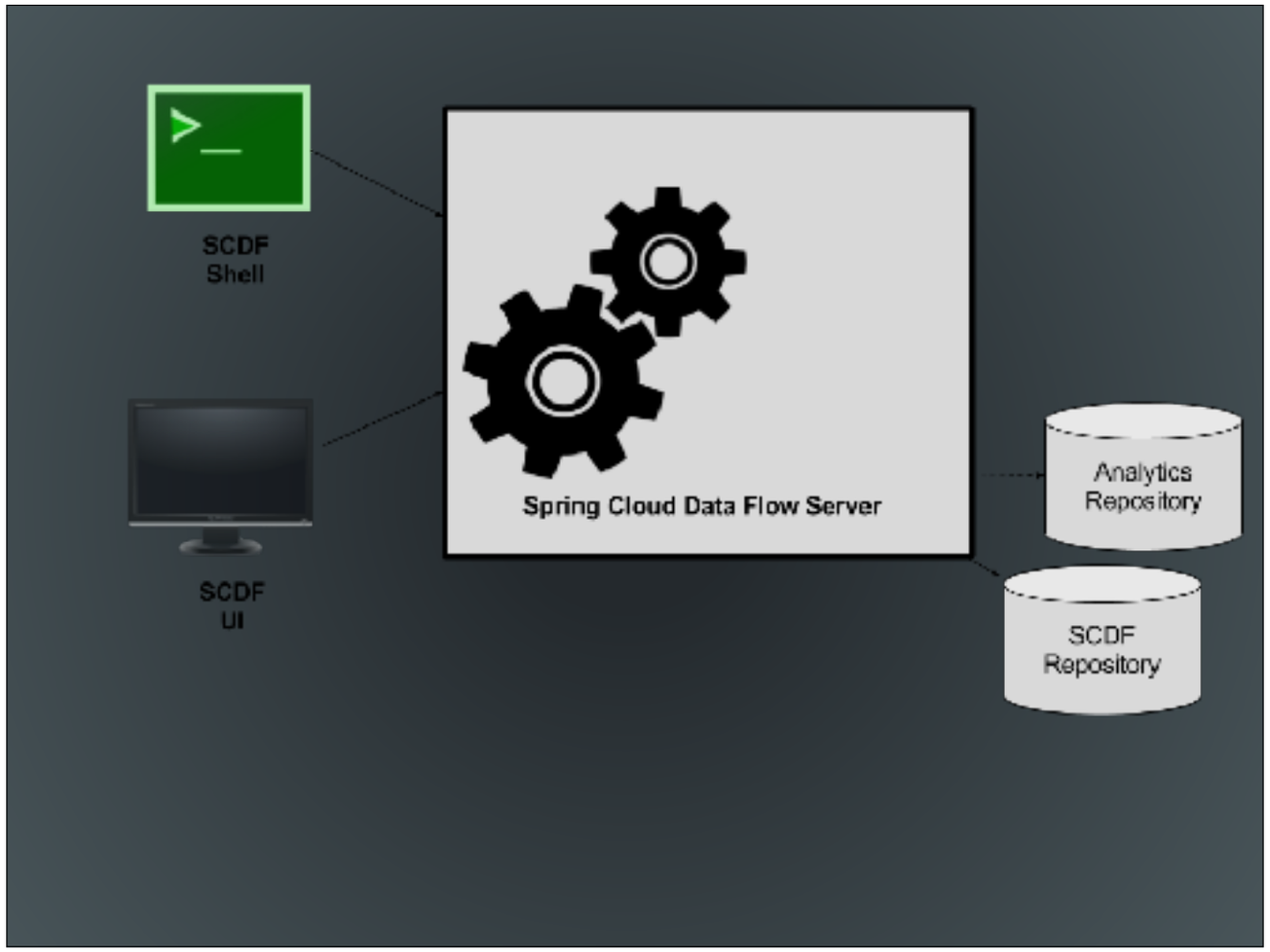
- Is an external Redis database that stores:
- Counts
- Aggregate-Counters
- Field Value Counters



SCDF UI

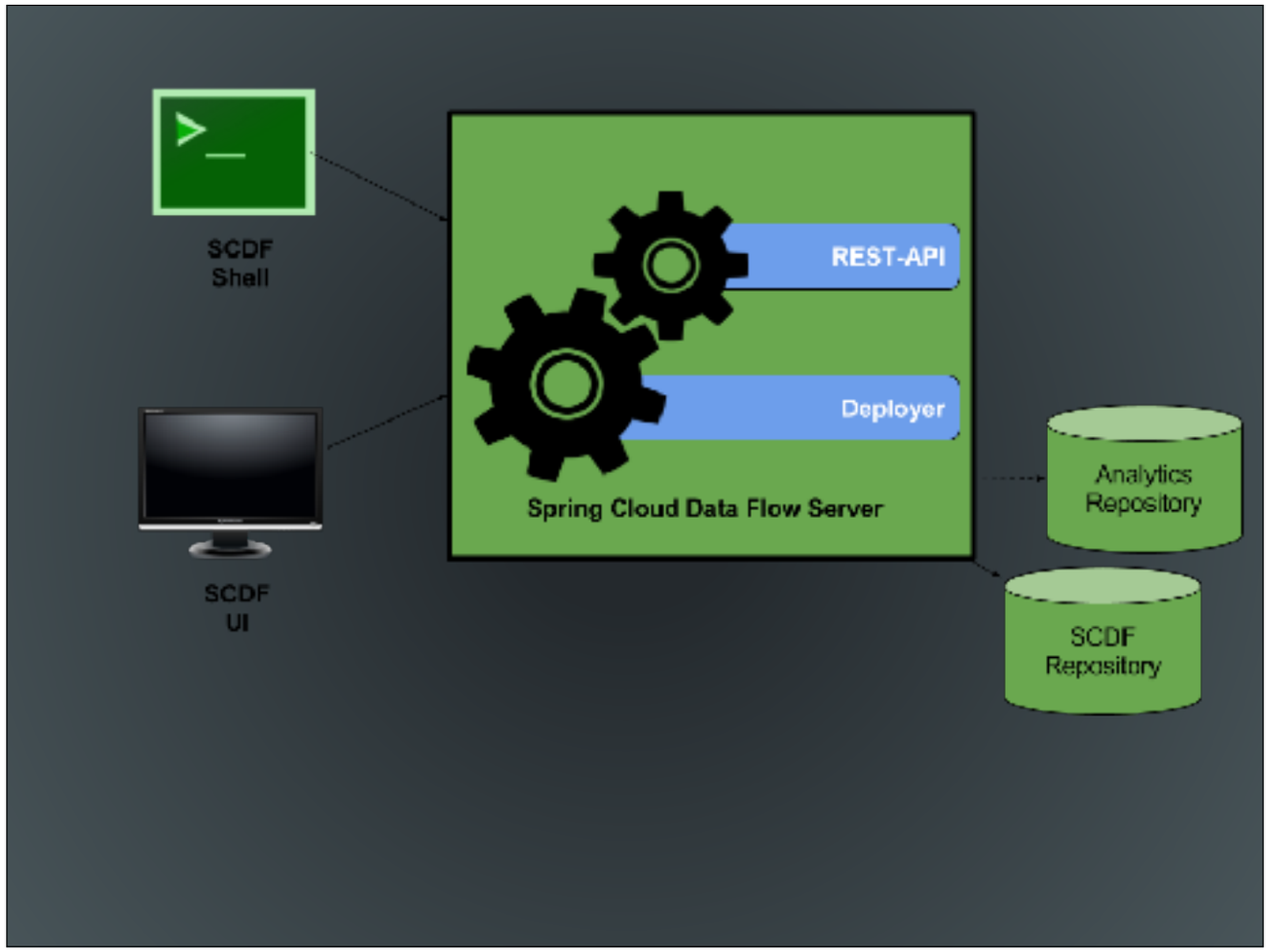
Offers an UI interface to the features offered by the RestfulAPI

- Accessible via the /dashboard endpoint ie. localhost:9393/dashboard
- Stream Creation, destruction and monitoring
- Task Creation, destruction and monitoring
- App registration
- Metrics montoring
- Job Monitoring



SCDF Shell

- Offers a command line interface to interface with the Restful API
- The shell has no real knowledge of the SCDF except what it gets from the Spring Cloud Data Flow Server
- Since it is a stand-alone app it can connect to any SCDF server that has its Restful API available for access
- By default it looks at localhost:9393/
-



Now we have the basic components that makeup a SCDF Server.

- Startup Data Flow Server
- Startup Shell
- Startup Rabbit

Security!

- Https
- Basic Authentication via LDAP or File based
- Single Sign On OAuth
  https://flic.kr/p/igAH3a

But, where are my apps?

- As we just discovered there are no apps
  - Spring Cloud Data Flow out of the box does not have a default set of apps that are available.
- You can register the apps that you want or
- You can import from our Spring Cloud App Starters if you wish to have a base set
  - app import --uri http://bit.ly/Avogadro-GA-stream-applications-rabbit-maven
- Now you can list the apps by executing an app list
- Get the details about a specific app i.e. app info <apptype>:appName
- Unregister the app. App unregister --name appName --type apptype
- App registration does not mean that the app has been pulled down to your server(when using https or maven).
  - It is only pulled into the maven repo or to file system when you deploy a stream, launch a task or execute app info

https://flic.kr/p/9zUCyn


```
app register --name http
--type source
--uri maven://
org.springframework.cloud.stream.app:http-
source-rabbit:1.1.1.RELEASE
```

Registration sample

- The registration command looks like the following: app register --name <yourappname> --type=<task,source,sink,processor> --uri=<thelocationofyourapp
- We currently support the following URI resources for looking up the app
    - File
    - Http
    - Maven

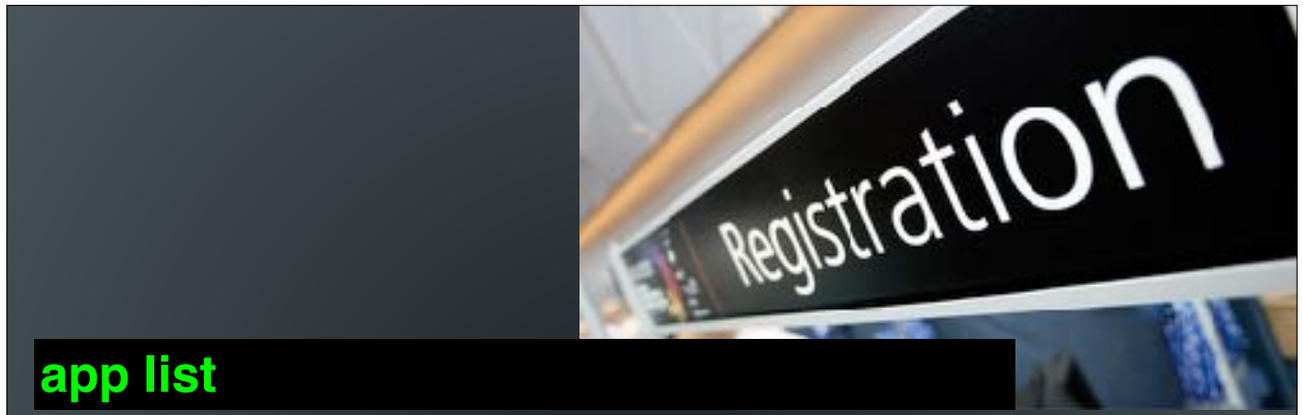
https://flic.kr/p/9YxwXp


```
app import
--uri http://bit.ly/Avogadro-GA-stream-
applications-rabbit-maven
```

I can register one app at a time or I can import them from a flat file or via http:

For example you can download our Spring Cloud App stream Starters app definitions for a basic set of apps from the shell:

- app import --uri http://bit.ly/Avogadro-GA-stream-applications-rabbit-maven



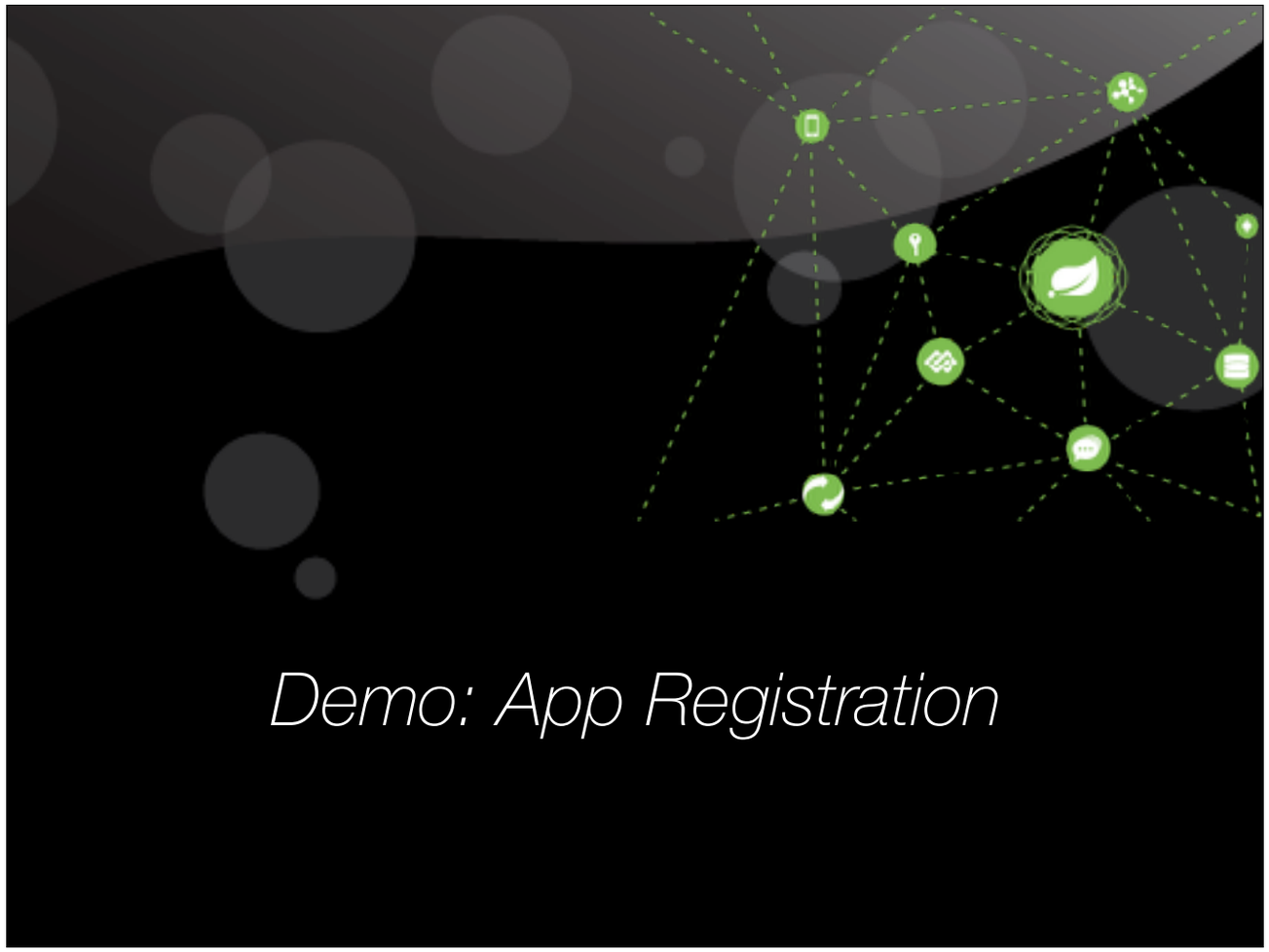

```
app list
```

```
app info --id source:http
```

```
app unregister --name http --type source
```

Once apps are registered we can exercise them a little bit.

- app list - obtain a list of the available apps
- app info - get detailed information about your application.
    - What properties does the app support
    - What is the apps resource location
    - Note: the first time the app is access either by deployment or app info, it will take longer if it has to pull it from a remote location.
- Finally Remove the app from the registry



- Import apps
- App list
- App info
- Create basic stream
- Deploy stream
- Show where results go