Ludion Documentation

Release 0.1.0

Samuel KORTAS

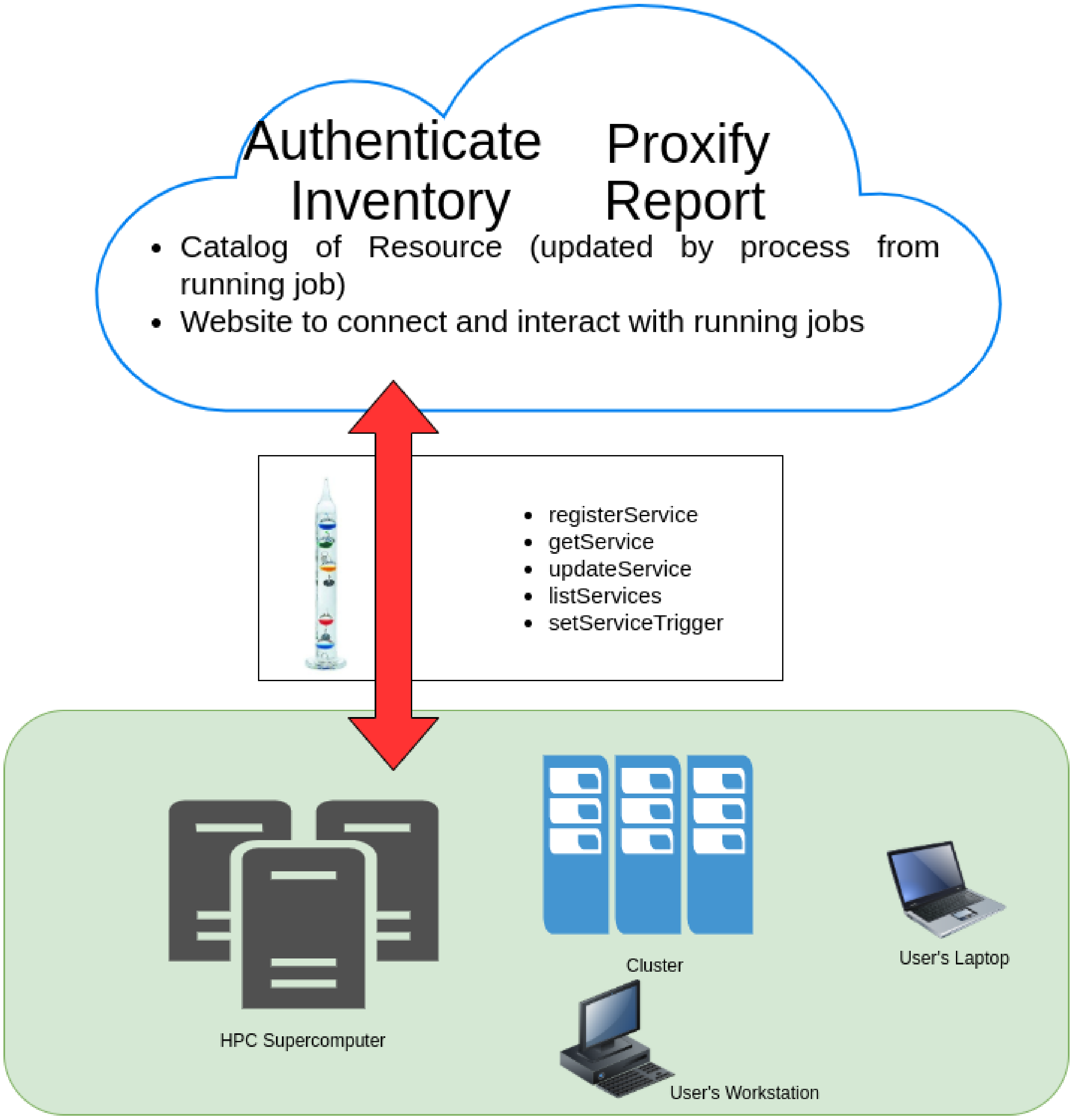
Nov 22, 2020

User Documentation

Ludion is the French word for "Cartesian Diver" a classic science experiment demonstrating the principle of Archimedes'principle. It consists of a diver trapped in a water bottle traveling down when a pressure is applied at the top of the enclosure and back up when it is released.

The movement of this diver portrays well the purpose of *Ludion* this framework which goal is to spread valuable information back and forth from HPC Resources to an easy to use web interface hosted in the AWS Cloud and vice versa.

Although all technical information can be found on the current website, the following [Article] might be of interest to understand the context of this work as it captures the motivations of *Ludion*'s development, its functional architecture, some detail of implementation as well as 10 use cases partially deployed at KAUST Supercomputing Laboratory.



CHAPTER 1

What is *Ludion*?

Developped by the KAUST Supercomputing Laboratory (KSL), *Ludion* is a service-oriented hybrid architecture, well-adapted to launch, monitor and steer interactive services spawned either on on-premise HPC resources, on user laptop and workstation or in the Cloud. Based on AWS serverless components, requiring no special priviledges to be deployed, it consists in a catalog of services and a dashboard hosted in the Cloud and a set of commands to install on the Resources to cover. From a running job, a user can register and publish his new service and any relevant data related on a centralized dashboard.

Developed by the KAUST Supercomputing Laboratory, *Ludion* is released as an Open Source Software under BSD Licence. It is available at http://github.com/samkos/ludion

1.1 Features

Ludion allows a user to:

- "publish" his own service via a centralized web interface and API. This service can be hosted either on-premise HPC, on his local workstation or in the cloud.
- dynamically update any information judged of interest about the published service
- make user-defined widget appear on the website in the view related to a given service. When clicked-on or filled and submitted, these widget trigger immediate action from the service hosted on resource.

CHAPTER 2

Installation

2.1 Requirements

Ludion does not require any super priviledge. It can be installed by a regular user as long as he possess an AWS account to install *Ludion* centralized services.

2.2 Distribution

Ludion is an open-source project distributed under the BSD 2-Clause "Simplified" License which means that many possibilities are offered to the end user including the fact to embed *Ludion* in one own software.

Its stable production branch is available via github at https://github.com/samkos/ludion where its latest production and development branch can be found

The most recently updated documentation can be browsed at http://ludion.readthedocs.io.

2.3 Installing *Ludion*

Ludion is composed of:

- a centralized dashboard and a GraphQL interface, hosted on a set of AWS serverless Resources, deployed thanks to *AWS Amplify*
- a set of scripts executable in a Unix shell to be installed on the connected Resources to connect to this centralized dashboard.

Current source is available on Github, use the following command to retrieve the latest stable version from the repository:

```
$ git clone https://github.com/samkos/ludion.git
```

2.4 Installing *Ludion* Centralized Services

Ludion relies on the following serverless AWS Components:

- 4 *DynamoDB* databases
- *AWS SES* to send mails,
- *Cognito user pool* to handle authentication of users that wish to connect to the website.
- *AWS Amplify* to deploy the dashboard and its corresponding *GraphQL* interface via *AWS Appsync*

Based on a *Cloud Formation* script, *Ludion* should be straightforward to deploy on AWS Cloud. We are still working on a fully automated installation, learning at the same time how to master *Cloud Formation* for this case.

We are presenting here a semi-automated installation using shell scripts and some amplify command that can not be scripted yet.

2.4.1 Prerequisites

This installation supposes that the current user

- has created an account on AWS
- has set up the AWS Email Service, SES in order to be able to send a mail from AWS

and that he installed on a local machine

- the aws cli (based on python)
- a recent version of nodeJS
- aws amplify (version>=4.32.1)

2.4.2 Deployment of the Dashboard in AWS cloud:

Let's deploy a version of *Ludion* that we will tag **prod**. Here are reproduced below the steps to install the centralized dashboard of *Ludion* built on serverless AWS components. At this stage, this steps are either automated either still manual. For the manual parts, accepting all the default choices is only required.

1. Clone the latest stable version of *Ludion* from Github:

```
$ git clone https://github.com/samkos/ludion.git LUDION_TEST
```

2. Initialize the amplify environment

```
$ cd LUDION_TEST/ludion
$ bash ../install/amplify_init.sh test

Note: It is recommended to run this command from the root of your app directory

For more information on AWS Profiles, see:
https://docs.aws.amazon.com/cli/latest/userguide/cli-multiple-profiles.html

Adding backend environment test to AWS Amplify Console app: d26x5q23er3ls4
 Initializing project in the cloud...

CREATE_IN_PROGRESS UnauthRole                AWS::IAM::Role            Thu Nov 12␣
↪2020 15:30:08 GMT+0300 (Arabian Standard Time)
```

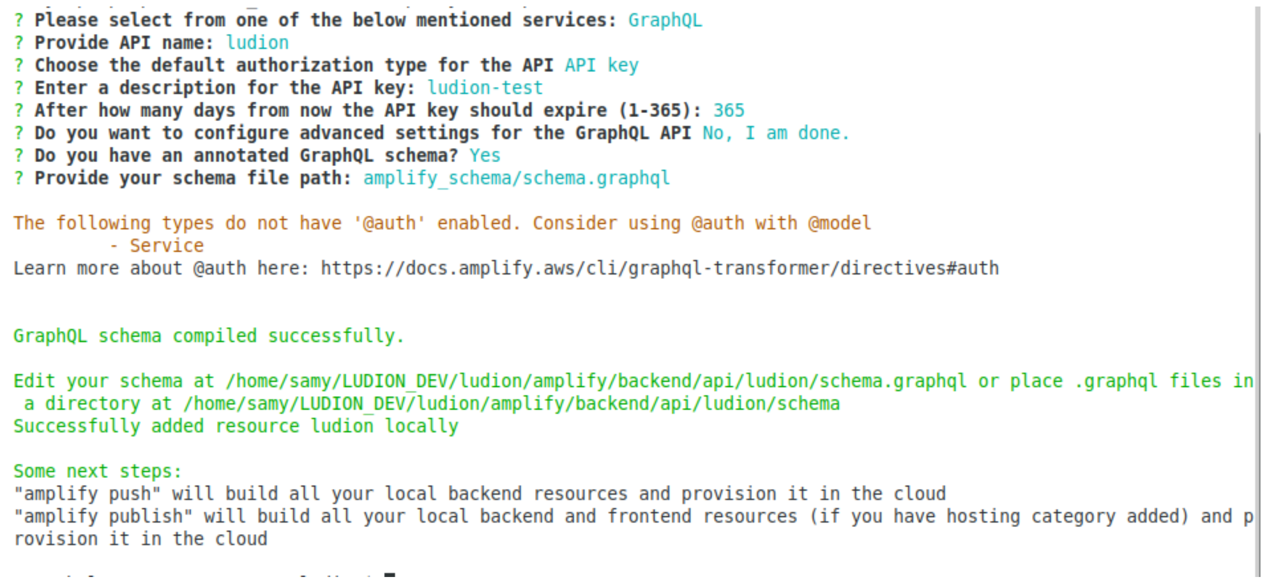
(continues on next page)

(continued from previous page)

```
CREATE_IN_PROGRESS AuthRole                    AWS::IAM::Role             Thu Nov 12␣
↪2020 15:30:08 GMT+0300 (Arabian Standard Time)
CREATE_IN_PROGRESS amplify-ludion-test-152955 AWS::CloudFormation::Stack Thu Nov 12␣
↪2020 15:30:03 GMT+0300 (Arabian Standard Time) User Initiated
 Initializing project in the cloud...

...

✓ Successfully created initial AWS cloud resources for deployments.
✓ Initialized provider successfully.
✓ All resources are updated in the cloud


Initialized your environment successfully.

Your project has been successfully initialized and connected to the cloud!

Some next steps:
"amplify status" will show you what you've added already and if it's locally␣
↪configured or deployed
"amplify add <category>" will allow you to add features like user login or a backend␣
↪API
"amplify push" will build all your local backend resources and provision it in the␣
↪cloud
"amplify console" to open the Amplify Console and view your project status
"amplify publish" will build all your local backend and frontend resources (if you␣
↪have hosting category added) and provision it in the cloud

Pro tip:
Try "amplify add api" to create a backend API and then "amplify publish" to deploy␣
↪everything
```

3. Add the GraphQL API, providing the model schema from amplify_schema/schema.graphql and setting the expiration time of the API key to 365 days not to have to change it too often.

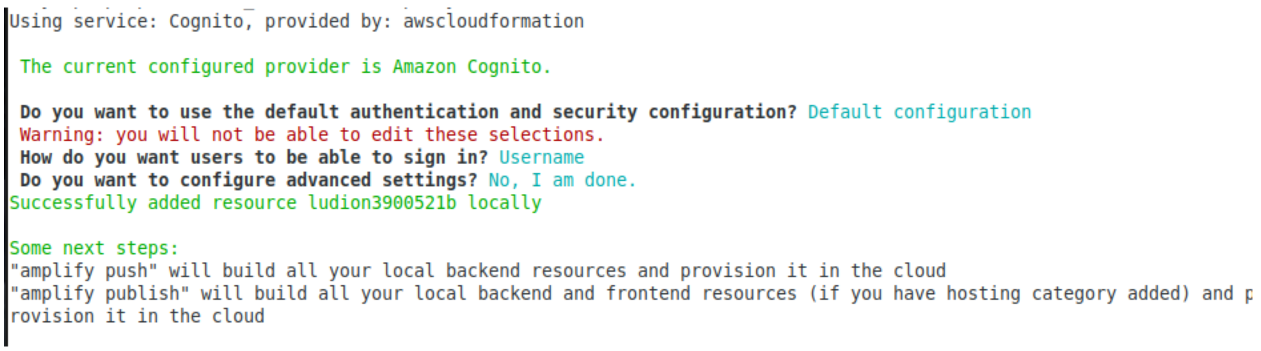
```
$ amplify add api
```



```
? Please select from one of the below mentioned services: GraphQL
? Provide API name: ludion
? Choose the default authorization type for the API API key
? Enter a description for the API key: ludion-test
? After how many days from now the API key should expire (1-365): 365
? Do you want to configure advanced settings for the GraphQL API No, I am done.
? Do you have an annotated GraphQL schema? Yes
? Provide your schema file path: amplify_schema/schema.graphql

The following types do not have '@auth' enabled. Consider using @auth with @model
         - Service
Learn more about @auth here: https://docs.amplify.aws/cli/graphql-transformer/directives#auth


GraphQL schema compiled successfully.

Edit your schema at /home/samy/LUDION_DEV/ludion/amplify/backend/api/ludion/schema.graphql or place .graphql files in
 a directory at /home/samy/LUDION_DEV/ludion/amplify/backend/api/ludion/schema
Successfully added resource ludion locally

Some next steps:
"amplify push" will build all your local backend resources and provision it in the cloud
"amplify publish" will build all your local backend and frontend resources (if you have hosting category added) and p
rovision it in the cloud
```

4. Link to an authentication via cognito user pool

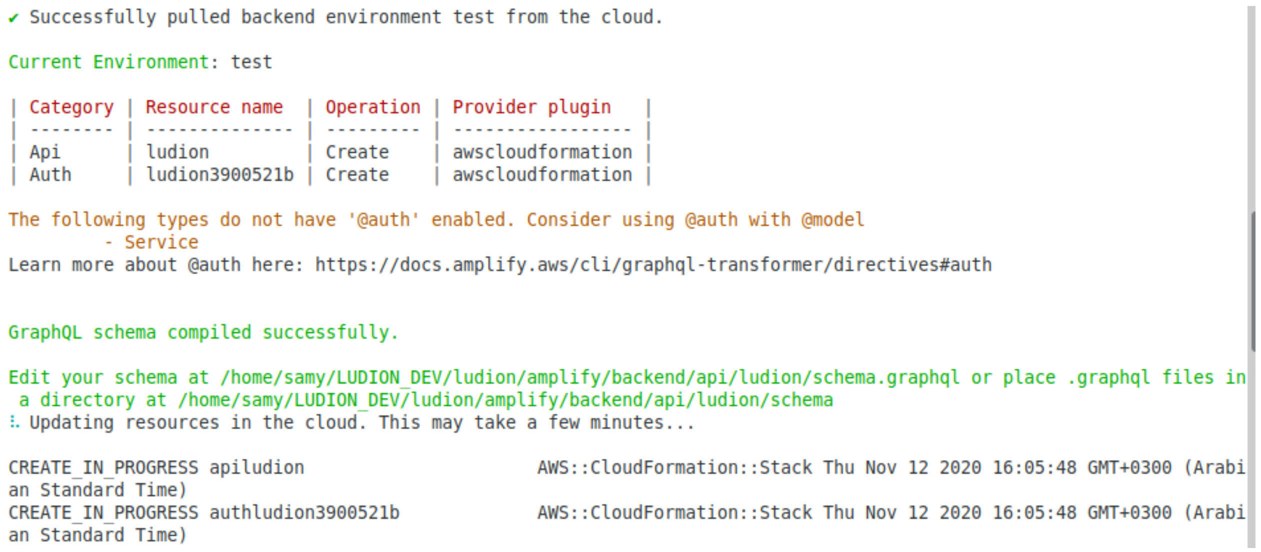
```
$ amplify add auth
```



```
Using service: Cognito, provided by: awscloudformation

 The current configured provider is Amazon Cognito.

 Do you want to use the default authentication and security configuration? Default configuration
 Warning: you will not be able to edit these selections.
 How do you want users to be able to sign in? Username
 Do you want to configure advanced settings? No, I am done.
Successfully added resource ludion3900521b locally

Some next steps:
"amplify push" will build all your local backend resources and provision it in the cloud
"amplify publish" will build all your local backend and frontend resources (if you have hosting category added) and p
rovision it in the cloud
```

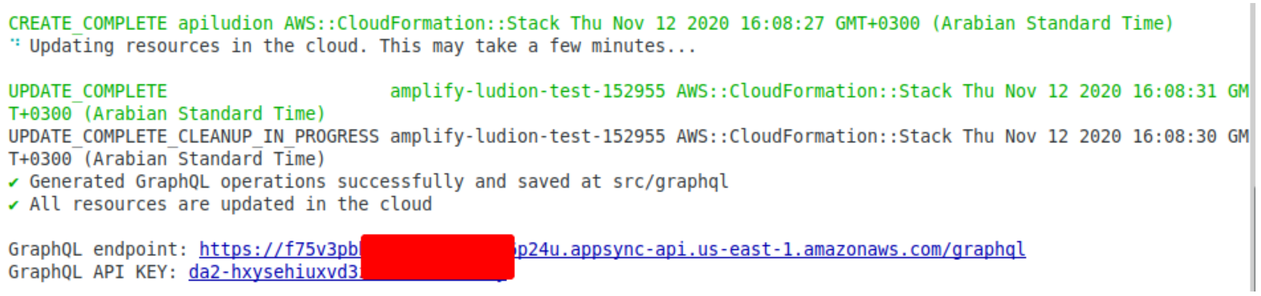
5. push the environment to the cloud

```
$ bash ../install/amplify_push.sh
```



```
✔ Successfully pulled backend environment test from the cloud.

Current Environment: test

| Category | Resource name  | Operation | Provider plugin   |
| -------- | -------------- | --------- | ----------------- |
| Api      | ludion         | Create    | awscloudformation |
| Auth     | ludion3900521b | Create    | awscloudformation |

The following types do not have '@auth' enabled. Consider using @auth with @model
         - Service
Learn more about @auth here: https://docs.amplify.aws/cli/graphql-transformer/directives#auth


GraphQL schema compiled successfully.

Edit your schema at /home/samy/LUDION_DEV/ludion/amplify/backend/api/ludion/schema.graphql or place .graphql files in
 a directory at /home/samy/LUDION_DEV/ludion/amplify/backend/api/ludion/schema
⠼ Updating resources in the cloud. This may take a few minutes...

CREATE_IN_PROGRESS apiludion                  AWS::CloudFormation::Stack Thu Nov 12 2020 16:05:48 GMT+0300 (Arabi
an Standard Time)
CREATE_IN_PROGRESS authludion3900521b         AWS::CloudFormation::Stack Thu Nov 12 2020 16:05:48 GMT+0300 (Arabi
an Standard Time)
```

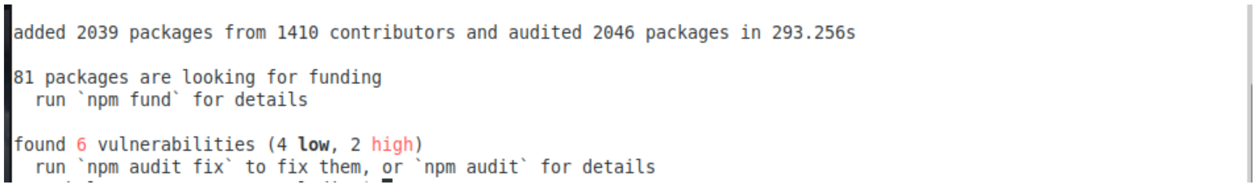
. . . Updating resources in the cloud. This may take a few minutes. . . . . .



```
CREATE_COMPLETE apiludion AWS::CloudFormation::Stack Thu Nov 12 2020 16:08:27 GMT+0300 (Arabian Standard Time)
⠙ Updating resources in the cloud. This may take a few minutes...

UPDATE_COMPLETE                    amplify-ludion-test-152955 AWS::CloudFormation::Stack Thu Nov 12 2020 16:08:31 GM
T+0300 (Arabian Standard Time)
UPDATE_COMPLETE_CLEANUP_IN_PROGRESS amplify-ludion-test-152955 AWS::CloudFormation::Stack Thu Nov 12 2020 16:08:30 GM
T+0300 (Arabian Standard Time)
✔ Generated GraphQL operations successfully and saved at src/graphql
✔ All resources are updated in the cloud

GraphQL endpoint: https://f75v3pb[redacted]p24u.appsync-api.us-east-1.amazonaws.com/graphql
GraphQL API KEY: da2-hxysehiuxvd3
```

6. create the website locally

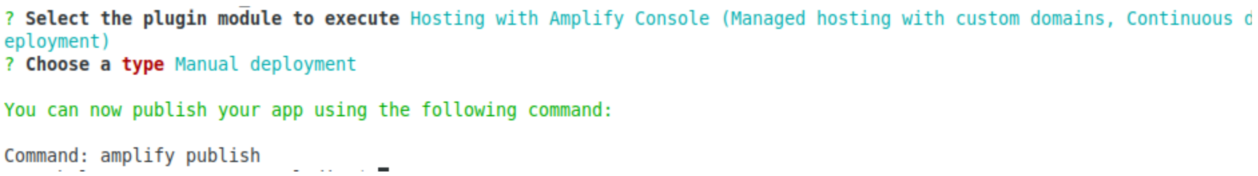
```
$ npm install
```



```
added 2039 packages from 1410 contributors and audited 2046 packages in 293.256s

81 packages are looking for funding
  run `npm fund` for details

found 6 vulnerabilities (4 low, 2 high)
  run `npm audit fix` to fix them, or `npm audit` for details
```

7. preparing the hosting place for the dashboard in the cloud

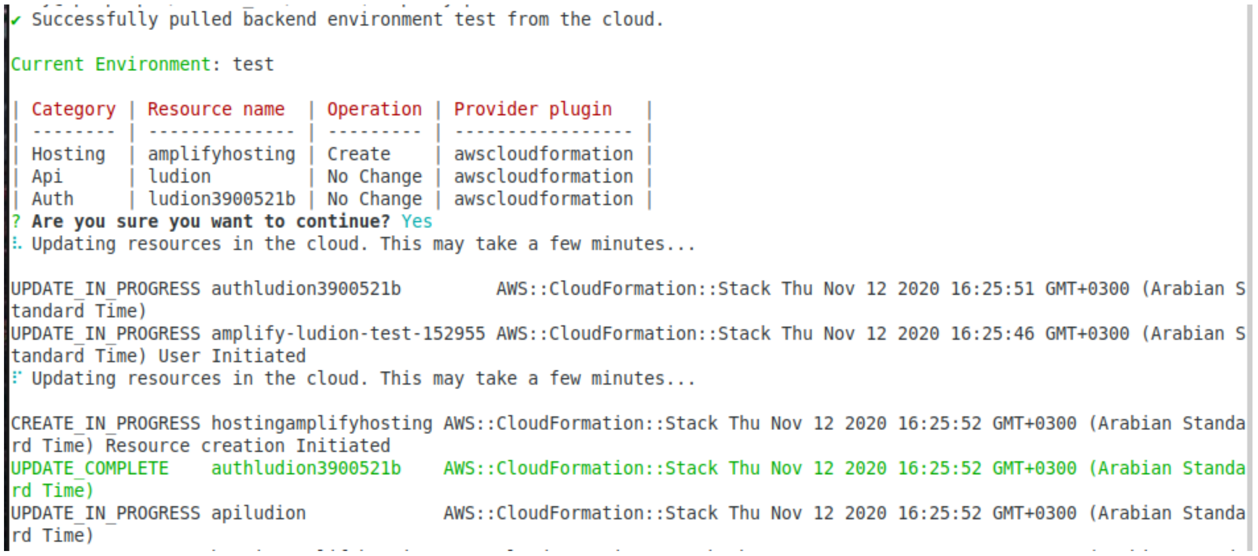
```
$ amplify hosting add
```



```
? Select the plugin module to execute Hosting with Amplify Console (Managed hosting with custom domains, Continuous d
eployment)
? Choose a type Manual deployment

You can now publish your app using the following command:

Command: amplify publish
```

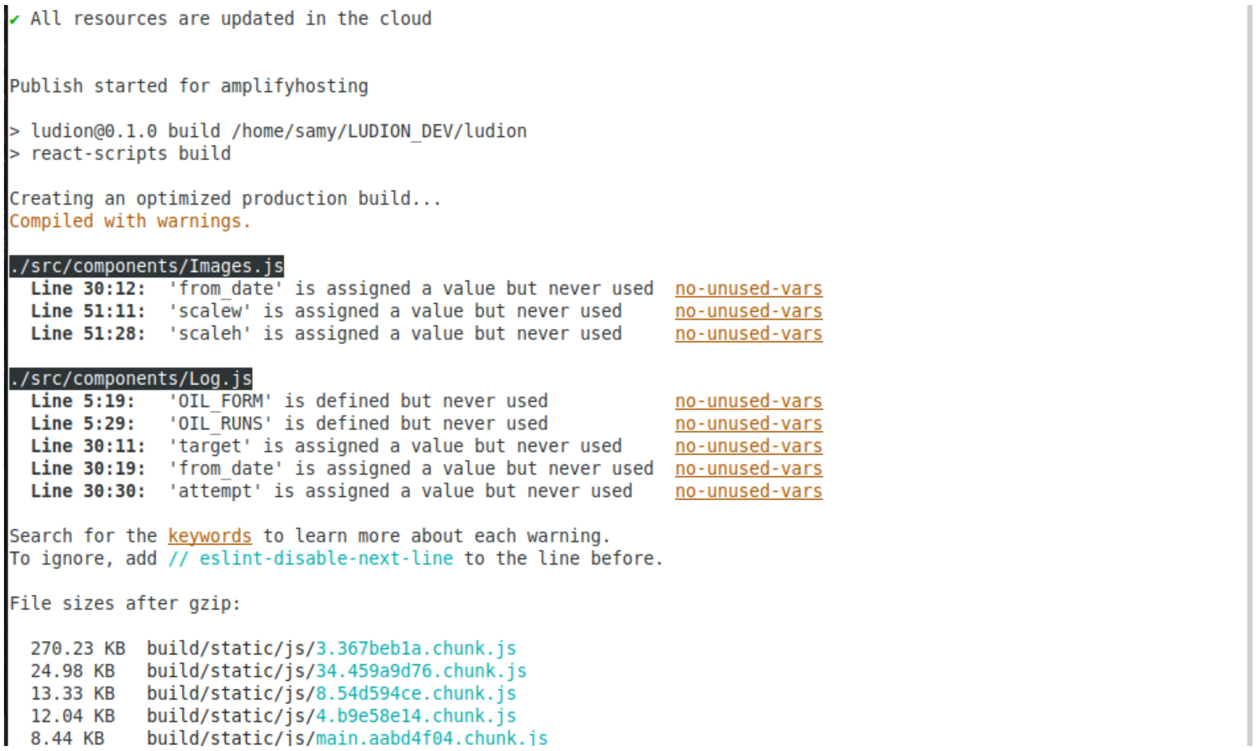
8. pushing it to the cloud

```
$ amplify publish
```



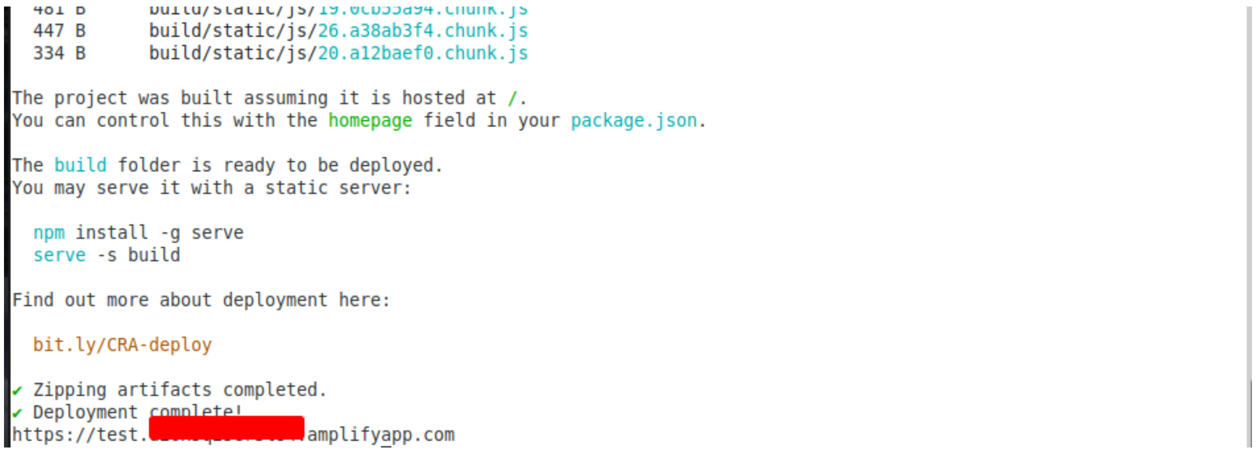
```
✔ Successfully pulled backend environment test from the cloud.

Current Environment: test

| Category | Resource name  | Operation | Provider plugin   |
| -------- | -------------- | --------- | ----------------- |
| Hosting  | amplifyhosting | Create    | awscloudformation |
| Api      | ludion         | No Change | awscloudformation |
| Auth     | ludion3900521b | No Change | awscloudformation |
? Are you sure you want to continue? Yes
⠼ Updating resources in the cloud. This may take a few minutes...

UPDATE_IN_PROGRESS authludion3900521b         AWS::CloudFormation::Stack Thu Nov 12 2020 16:25:51 GMT+0300 (Arabian S
tandard Time)
UPDATE_IN_PROGRESS amplify-ludion-test-152955 AWS::CloudFormation::Stack Thu Nov 12 2020 16:25:46 GMT+0300 (Arabian S
tandard Time) User Initiated
⠙ Updating resources in the cloud. This may take a few minutes...

CREATE_IN_PROGRESS hostingamplifyhosting AWS::CloudFormation::Stack Thu Nov 12 2020 16:25:52 GMT+0300 (Arabian Standa
rd Time) Resource creation Initiated
UPDATE_COMPLETE    authludion3900521b    AWS::CloudFormation::Stack Thu Nov 12 2020 16:25:52 GMT+0300 (Arabian Standa
rd Time)
UPDATE_IN_PROGRESS apiludion             AWS::CloudFormation::Stack Thu Nov 12 2020 16:25:52 GMT+0300 (Arabian Standa
rd Time)
```

continuing



```
✔ All resources are updated in the cloud


Publish started for amplifyhosting

> ludion@0.1.0 build /home/samy/LUDION_DEV/ludion
> react-scripts build

Creating an optimized production build...
Compiled with warnings.

./src/components/Images.js
  Line 30:12:  'from_date' is assigned a value but never used  no-unused-vars
  Line 51:11:  'scalew' is assigned a value but never used     no-unused-vars
  Line 51:28:  'scaleh' is assigned a value but never used     no-unused-vars

./src/components/Log.js
  Line 5:19:   'OIL_FORM' is defined but never used            no-unused-vars
  Line 5:29:   'OIL_RUNS' is defined but never used            no-unused-vars
  Line 30:11:  'target' is assigned a value but never used     no-unused-vars
  Line 30:19:  'from_date' is assigned a value but never used  no-unused-vars
  Line 30:30:  'attempt' is assigned a value but never used    no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.

File sizes after gzip:

  270.23 KB  build/static/js/3.367beb1a.chunk.js
  24.98 KB   build/static/js/34.459a9d76.chunk.js
  13.33 KB   build/static/js/8.54d594ce.chunk.js
  12.04 KB   build/static/js/4.b9e58e14.chunk.js
  8.44 KB    build/static/js/main.aabd4f04.chunk.js
```

continuing



```
  481 B      build/static/js/19.0cb55a94.chunk.js
  447 B      build/static/js/26.a38ab3f4.chunk.js
  334 B      build/static/js/20.a12baef0.chunk.js

The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:

  npm install -g serve
  serve -s build

Find out more about deployment here:

  bit.ly/CRA-deploy

✔ Zipping artifacts completed.
✔ Deployment complete!
https://test.[redacted]amplifyapp.com
```

9. Installing *Ludion* local components

```
$ bash ../install/ludion_cli_configure.sh
```

At this point the ludion cli unix commands should be fully configured in the *../API/unix/* directory. Adding this directory to *PATH* variable completes the installation of *Ludion*

CHAPTER 3

Ludion's API

At this stage, *Ludion* API is still under development. For now, its public API is only available as Unix commands and NodeJS, Python and Dart languages are expected to be supported in the near future.

Currently under development and test, the following Unix commands are planned to be released with the first stable release of *ludion* expected by November 2020:

- **registerService**, to register a new service,
- **getService**, to get details about a service already deployed,
- **updateService**, to update values relative to a service expected to be published on **Ludion**'s dashboard,
- **listServices**, to list all available services,
- **setServiceTrigger**, to establish a graphical widget in the Ludion's dashboard in the view related to the current service and triggers its activation to an action executed immediately in the corresponding job

their use will be detailed here.

3.1 registerService

Purpose: register a service in Ludion

Usage:

```
registerService.js --service [serviceName] --instance [instanceName]
```

Options:

```
--version         Show version number                                 [boolean]
-s, --service     Name of the service                                [required]
-i, --instance    Name of the instance                               [required]
-h, --help        Show help                                           [boolean]
-u, --user        given user | all                               [ ADMIN ONLY ]
-Z, --not-admin   act as regular user, abolish admin privileges  [ ADMIN ONLY ]
```

(continues on next page)

(continued from previous page)

```
-e, --endpoint   Endpoint
-d, --debug      Adds debug trace
-j, --job        job #
```

Example:

```
registerService.js -s Jupyter -i myBook

  -> launch a Jupyter note book service. It will appear in the dashboard
     as the instance *myBook* of the service *Jupyter*.
```

3.2 getService

Purpose: get current status and parameters for a given service registered in Ludion

Usage:

```
getService --service <serviceName> --instance <instanceName>
           [ --parameter param1[,param2,..] | --all-parameters ]
```

Options:

```
--version                Show version number                               [boolean]
-s, --service            Name of the service
-i, --instance           Name of the instance
-h, --help               Show help                                         [boolean]
-u, --user               given user | all                                  [ ADMIN ONLY ]
-Z, --not-admin          act as regular user, abolish admin privileges     [ ADMIN ONLY ]
-d, --debug              Adds debug trace
-a, --all-parameters     returns the value of all parameters
-p, --parameter          return the value of parameter listed
```

Examples:

```
getService.js -s Jupyter -i myBook       -> returns the current status of the␣
↪instance *myBook*
                                            of the service *Jupyter*


getService.js -s Jupyter -i myBook  -a   -> returns all parameter of the instance␣
↪*myBook*
                                            of the service *Jupyter*
```

3.3 updateService

Purpose: update status and/or parameters for a given service registered in Ludion

Usage:

```
updateService --service <serviceName> --instance <instanceName>
              [ --param1 value1 [ --param2 value2 ...] ]
```

Options:

```
--version        Show version number                                [boolean]
-s, --service    Name of the service
-i, --instance   Name of the instance
-h, --help       Show help                                          [boolean]
-u, --user       given user | all                              [ ADMIN ONLY ]
-Z, --not-admin  act as regular user, abolish admin privileges  [ ADMIN ONLY ]
-d, --debug      Adds debug trace
```

Examples:

```
updateService.js -s Jupyter -i myBook  -x 1 -y 2
-> set to 1 and 2  the resepctive parameters x and y of a
   the instance *myBook* of the service *Jupyter*
```

3.4 listServices

Purpose: list services registered in Ludion

Usage: :: listServices.js [–long] [–json]

[–service <serviceName>] [–instance <instanceName>]

Options:

```
--version        Show version number                                [boolean]
-s, --service    Name of the service
-i, --instance   Name of the instance
-h, --help       Show help                                          [boolean]
-u, --user       given user | all                              [ ADMIN ONLY ]
-Z, --not-admin  act as regular user, abolish admin privileges  [ ADMIN ONLY ]
-l, --long       long format
-j, --json       json format
-d, --debug      Adds debug trace
```

Examples:

```
listServices.js -u johndoe  -> list all services belonging to user johndoe
```

3.5 setServiceTrigger

Purpose: connect a widget of a service in Ludion to a local script

Usage:

```
setServiceTrigger --service <serviceName> --instance <instanceName>
                  --widget <widget_type> --label xxx --calls <script>
```

Options:

```
--version        Show version number                                [boolean]
-s, --service    Name of the service
-i, --instance   Name of the instance
-h, --help       Show help                                          [boolean]
```

(continues on next page)

(continued from previous page)

```
-u, --user       given user | all                                 [ ADMIN ONLY ]
-Z, --not-admin  act as regular user, abolish admin privileges  [ ADMIN ONLY ]
-w, --widget     Type of widget to add
-l, --label      Widget configuration parameters
-c, --calls      Absolute path to script to trigger
-d, --debug      Adds debug trace
```

Examples:

```
setServiceTrigger.js -s Jupyter -i        -> a click on Button "Save me"
myBook --widget Button --label "Save me"  triggers a call to run_save.sh
--calls run_save.sh
```

CHAPTER 4

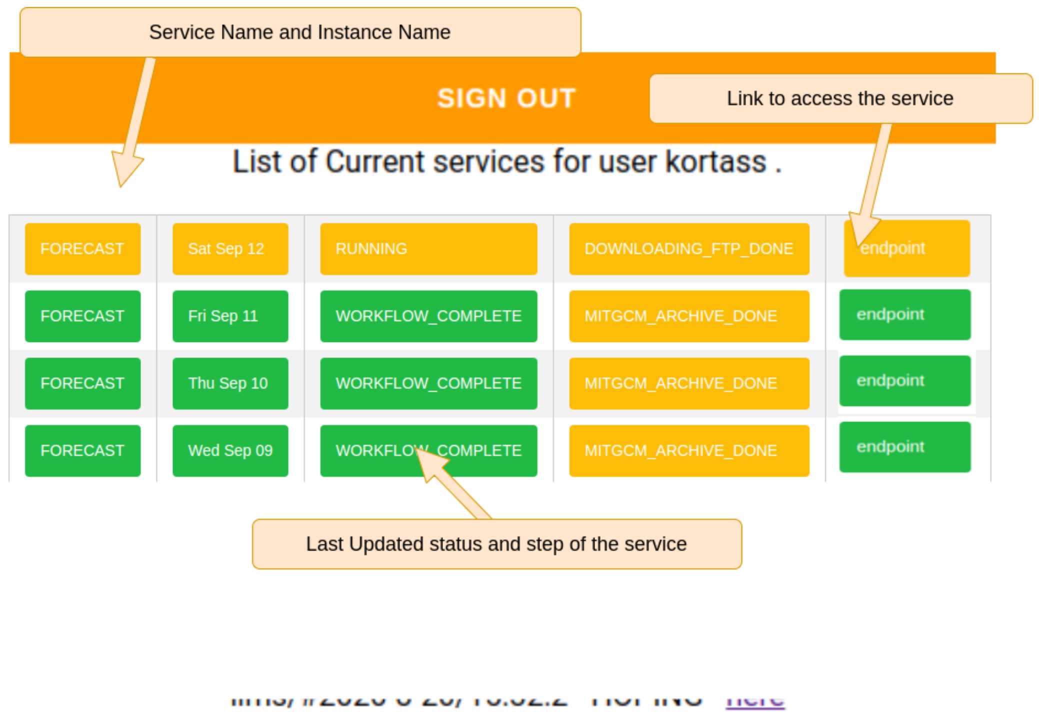
Typical use case: spawning a Jupyter Notebook

Let's take the example of a Jupyter Notebook to be deployed from a node of a cluster named *Ibex*. Let's assume that we are already logged in on a node of this cluster, *Ibex_000xxx* and have checked for the first port *Port_nnn* available and started a Jupyter Notebook server responding on this port. We also eventually set up a random password *Password_yyy* to secure access to the Notebook.

The fist step consists in registering Jupyter Notebook as the instance *example* of the service we name *JupyterNotebook*:

```
$ registerService --service JupyterNotebook
                  --instance example \
                  --endpoint http://Ibex_000xxx:Port_nnn
                  --password Password_yyy
service JupyterNotebook:example registered
```

In the cloud, this newly registered service adds a new line in *Ludion* centralized database, and triggers the update of any browser pointing to *Ludion*'s Dashboad showing that a service JupyterNotebook is now ready for the user to access at the address *http://Ibex_000xxx:Port_nnn* that appears as a clickable link using the password *Password_yyy* displayed with all other parameters of the service when clicking on the service.


Service Name and Instance Name
SIGN OUT
Link to access the service
Last Updated status and step of the service

List of Current services for user kortass .

FORECAST	Sat Sep 12	RUNNING	DOWNLOADING_FTP_DONE	endpoint
FORECAST	Fri Sep 11	WORKFLOW_COMPLETE	MITGCM_ARCHIVE_DONE	endpoint
FORECAST	Thu Sep 10	WORKFLOW_COMPLETE	MITGCM_ARCHIVE_DONE	endpoint
FORECAST	Wed Sep 09	WORKFLOW_COMPLETE	MITGCM_ARCHIVE_DONE	endpoint

Once the service is registered, any parameter can be updated thanks to the following command:

```
$ updateService --service JupyterNotebook
                --instance example
                --status RUNNING
                --step step_0.1
service JupyterNotebook:example updated successfully
```

These parameters are immediately updated in the centralized database and on the *Ludion* dashboard.

One can also retrieve given parameters of a given service with the command:

```
$ getService --service JupyterNotebook
             --instance example
             --parameters "endpoint,status,login,password,x1"
{ service  : "JupyterNotebook",
  instance : "example",
  endpoint : "Ibex_000123:2030",
  status   : "RUNNING"
  login    : "JupyterNotebook",
  password : "Password_yyy"
}
```

More briefly, this command can be called with no parameter to get only the status of the service:

```
$ getService --service JupyterNotebook
             --instance example
RUNNING
```

Or, to get all parameters with –all-parameters options

```
$ getService --service JupyterNotebook
             --instance example
             --all-parameters
{ service    : "JupyterNotebook",
  instance   : "example",
  id         : "JupyterNotebook_example_1141442334333311",
  description: "Jupyter service",
  user       : "user_login",
  machine    : "Ibex",
  endpoint   : "Ibex_000123:2030",
  status     : "COMPLETE",
  step       : "step_0.1",
  password   : "Password_yyy",
  createdAt  : "2020-11-11 14:00:00",
  updatedAt  : "2020-11-11 14:10:410",
  jobid      " "012121544"
}
```

Bibliography

[Article] Towards an HPC Service Oriented Hybrid Cloud Architecture Designed for Interactive Workflows, Samuel KORTAS & Moshin SHAIKH, submitted in September 2020, available on request.