Container Based Task Execution Workflow for Airavata

User Guide

Dimuthu Upeksha

Introduction

This user guide explains how to deploy an Airavata container based workflow in a Kubernetes setup and manage it using a Web Console which is analogous to PGA

Source code for all the components can be found from
https://github.com/DImuthuUpe/airavata/tree/master/sandbox/airavata-kubernetes

Prerequisites

1. Docker installed in your machine
2. MySQL database
3. Kafka Broker (one instance is enough)
4. Kubernetes Installation (To developments purposes, we can use a Minikube distribution which is a developer version of Kubernetes)

Configure MySQL database

Create a database named "airavata"
Create a user and assign privileges of database to the user

```
CREATE DATABASE airavata;
CREATE USER airavata-admin@% IDENTIFIED BY 'password';
GRANT ALL ON airavata.* TO airavata-admin@'%';
FLUSH PRIVILEGES;
```

Make sure that MySQL service is accessible from other hosts other than localhost. This can be done by adding adding configuration to /etc/my.cnf file. To apply the changes, you have to restart MySQL the service

```
[mysqld]
bind-address        = 0.0.0.0
```

Set up a Kafka Broker

Instructions to create a simple Kafka deployment can be found from here
https://kafka.apache.org/quickstart

Using Kafka cli tool, create following topics (change the zookeeper connect string according to your configuration)

```
bin/kafka-topics.sh --create --zookeeper localhost:2199
--replication-factor 1 --partitions 100 --topic airavata-launch

bin/kafka-topics.sh --create --zookeeper localhost:2199
--replication-factor 1 --partitions 100 --topic airavata-scheduler

bin/kafka-topics.sh --create --zookeeper localhost:2199
--replication-factor 1 --partitions 100 --topic
airavata-task-ingress-staging

bin/kafka-topics.sh --create --zookeeper localhost:2199
--replication-factor 1 --partitions 100 --topic
airavata-task-egress-staging

bin/kafka-topics.sh --create --zookeeper localhost:2199
--replication-factor 1 --partitions 100 --topic
airavata-task-env-setup

bin/kafka-topics.sh --create --zookeeper localhost:2199
--replication-factor 1 --partitions 100 --topic
airavata-task-env-cleanup

bin/kafka-topics.sh --create --zookeeper localhost:2199
--replication-factor 1 --partitions 100 --topic
airavata-task-job-submission
```

Install Minikube

Follow this guide to install Minikube locally
https://kubernetes.io/docs/tasks/tools/install-minikube.

Note: To install Minikube, you should have virtualbox installed in your machine

1. Start Minikube

```
minikube start
```

2. Verify whether Minikube has been successfully configured

```
kubectl get nodes
```

It should show an output like this

```
NAME        STATUS    ROLES     AGE       VERSION
minikube    Ready     <none>    3m        v1.8.0
```

Install Airavata Microservices

Scripts to install Airavata on Kubernetes can be found from. You need these scripts copied to your machine
https://github.com/DImuthuUpe/airavata/tree/master/sandbox/airavata-kubernetes/scripts/k8s

Prior to the installation, do following changes to those scripts

1. Change ip of db-service.yml to your database host name
https://github.com/DImuthuUpe/airavata/blob/master/sandbox/airavata-kubernetes/scripts/k8s/db-service.yml

```
subsets:
  - addresses:
      - ip: 192.168.1.114
```

2. Change ip of kafka-service.yml to your Kafka broker host name
https://github.com/DImuthuUpe/airavata/blob/master/sandbox/airavata-kubernetes/scripts/k8s/kafka-service.yml

```
subsets:
  - addresses:
      - ip: 192.168.1.114
```

3. Set the database username and password in api-server-dep.yml according to your environment
https://github.com/DImuthuUpe/airavata/blob/master/sandbox/airavata-kubernetes/scripts/k8s/api-server/api-server-dep.yml

4. Run installation scripts in following order

```
Dimuthus-MacBook-Pro:scripts dimuthu$ kubectl create -f
k8s/db-service.yml
service "db" created
endpoints "db" created

Dimuthus-MacBook-Pro:scripts dimuthu$ kubectl create -f
k8s/kafka-service.yml
service "kafka" created
endpoints "kafka" created
```

```
Dimuthus-MacBook-Pro:scripts dimuthu$ kubectl create -f
k8s/api-server/api-server-dep.yml
deployment "api-server" created

Dimuthus-MacBook-Pro:scripts dimuthu$ kubectl create -f
k8s/api-server/api-server-svc.yml
service "api-server" created

Dimuthus-MacBook-Pro:scripts dimuthu$ kubectl create -f
k8s/event-sink/event-sink-dep.yml
deployment "event-sink" created

Dimuthus-MacBook-Pro:scripts dimuthu$ kubectl create -f
k8s/task-scheduler/task-secheduler-dep.yml
deployment "task-scheduler" created

Dimuthus-MacBook-Pro:scripts dimuthu$ kubectl create -f
k8s/workflow-generator/workflow-generator-dep.yml
deployment "workflow-generator" created

Dimuthus-MacBook-Pro:scripts dimuthu$ kubectl create -f
k8s/tasks/egress-staging-task/egress-staging-task-dep.yml
deployment "egress-staging-task" created

Dimuthus-MacBook-Pro:scripts dimuthu$ kubectl create -f
k8s/tasks/env-setup-task/env-setup-task-dep.yml
deployment "env-setup-task" created

Dimuthus-MacBook-Pro:scripts dimuthu$ kubectl create -f
k8s/tasks/env-cleanup-task/env-cleanup-task-dep.yml
deployment "env-cleanup-task" created

Dimuthus-MacBook-Pro:scripts dimuthu$ kubectl create -f
k8s/tasks/ingress-staging-task/ingress-staging-task-dep.yml
deployment "ingress-staging-task" created

Dimuthus-MacBook-Pro:scripts dimuthu$ kubectl create -f
k8s/tasks/job-submission-task/job-submission-task-dep.yml
deployment "job-submission-task" created
```

This will take a while to download all the Docker images from the DockerHub

Run following command to view the status of all the pods.

```
kubectl get pods
```

If all pods are in running state as below, continue to next step

```
NAME                                        READY     STATUS    RESTARTS   AGE
api-server-697579c6d6-2xcr9                 1/1       Running   0          6m
egress-staging-task-654d69f88d-2zdp2        1/1       Running   0          5m
env-cleanup-task-5bc5888c5-599br            1/1       Running   0          3m
env-setup-task-75cccdd7d8-65wsq             1/1       Running   0          4m
event-sink-6c8b6b467-5rfdx                  1/1       Running   0          6m
ingress-staging-task-569b754b4d-p86tk       1/1       Running   0          2m
job-submission-task-758bd7b757-67qsj        1/1       Running   0          1m
task-scheduler-744d79db74-kcd9b             1/1       Running   0          5m
workflow-generator-7bfc9c9c9d-mct5z         1/1       Running   0          5m
```

5. Get the ip address of Minikube

```
Dimuthus-MacBook-Pro:~ dimuthu$ minikube ip
192.168.99.100
```

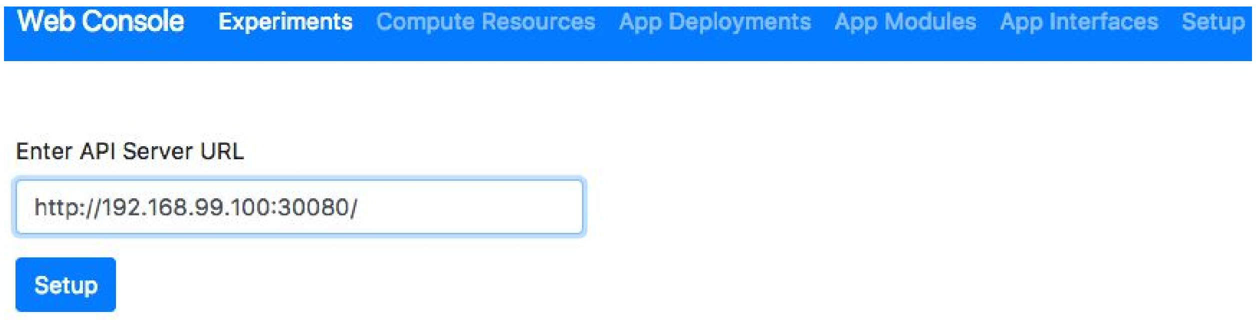
6. Launch the Web Console to submit and monitor experiments

```
docker run -it -p 80:80 dimuthuupe/airavata-console:v1.0
```

Using Web Console to create, launch and monitor workflows

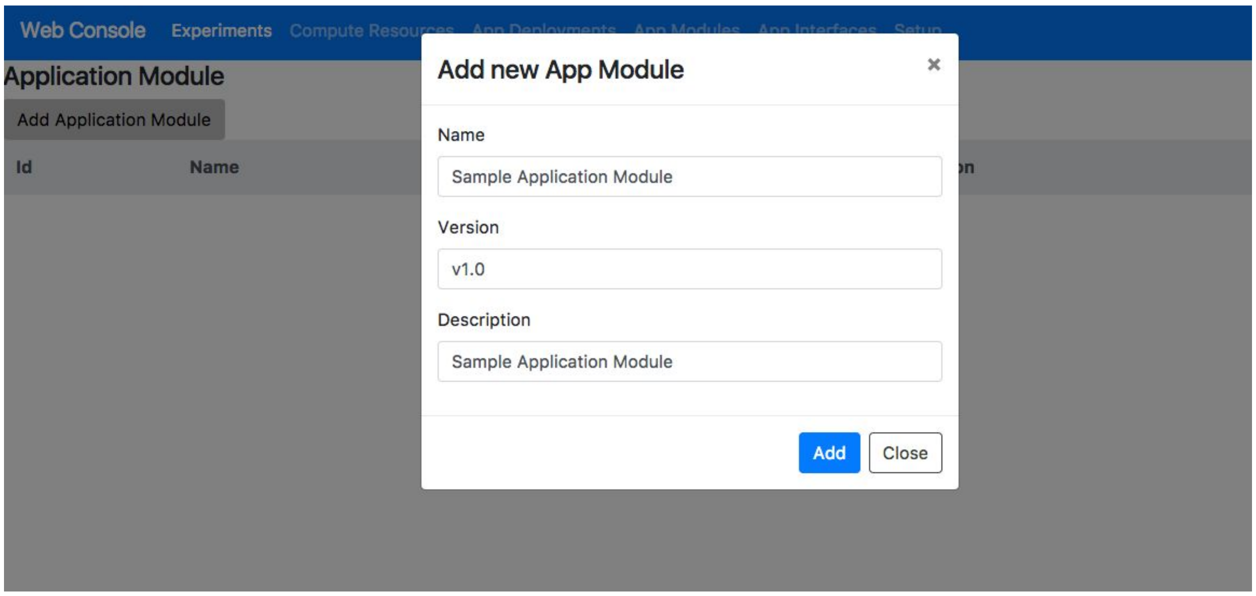
1. Goto http://localhost

2. Click Setup
Enter http://minikube-ip:30080 as the API Server URL



http://192.168.99.100:30080/

3. Go To App Modules and Create a new App Module

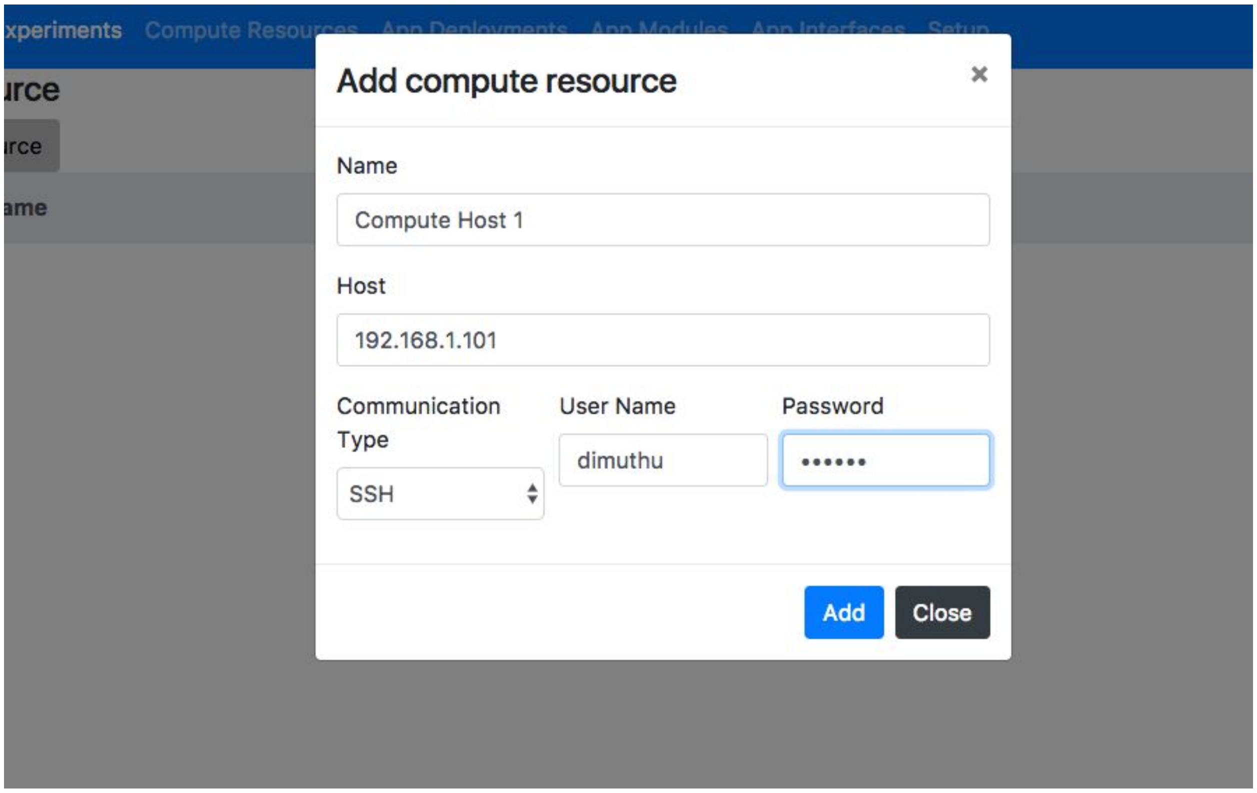


Name: Sample Application Module

Version: v1.0

Description: Sample Application Module

4. Add a compute resource. Currently this supports to SSH username password based authentication. So provide the user and password of target compute host



Add compute resource

Name: Compute Host 1

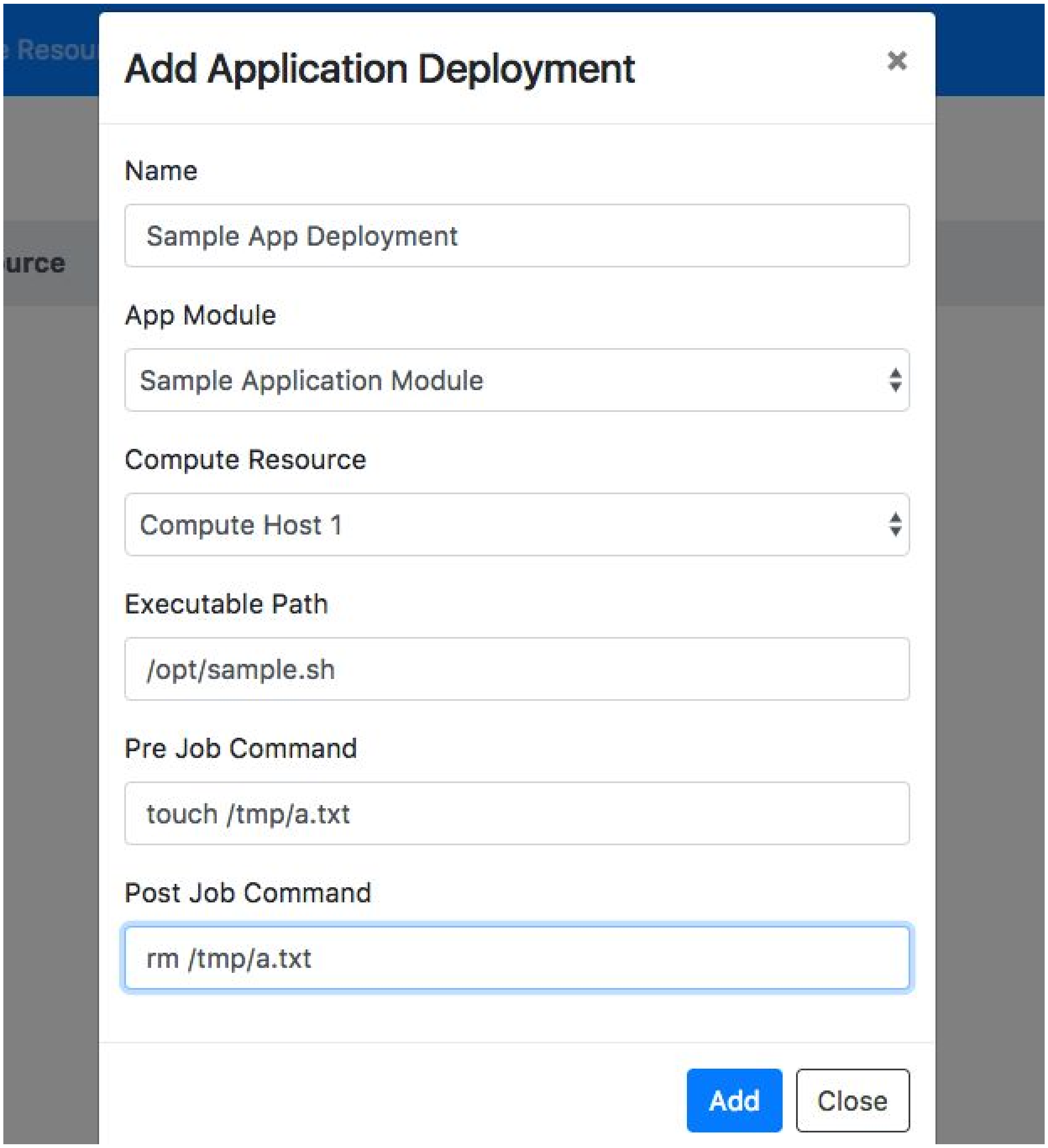
Host: 192.168.1.101

Communication Type: SSH

User Name: dimuthu

Password: ••••••

5. Add a Application Deployment. Select previously created App Module and Compute Host. Add executable path, pre job command and post job command accordingly



Add Application Deployment

Name: Sample App Deployment

App Module: Sample Application Module

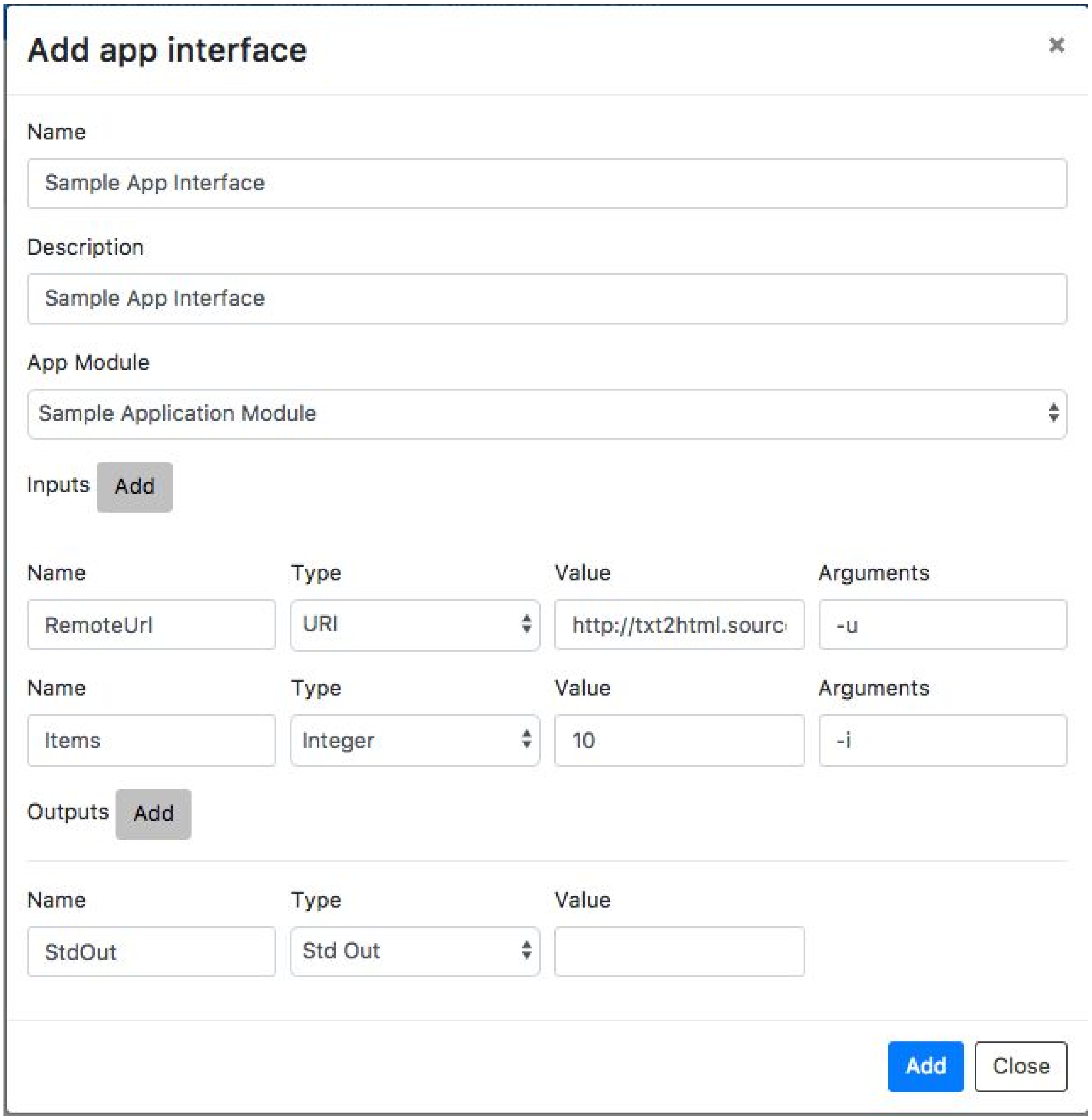
Compute Resource: Compute Host 1

Executable Path: /opt/sample.sh

Pre Job Command: touch /tmp/a.txt

Post Job Command: rm /tmp/a.txt

6. Add an Application interface. Select previously created Application module and configure inputs and outputs



Add app interface

Name: Sample App Interface

Description: Sample App Interface

App Module: Sample Application Module

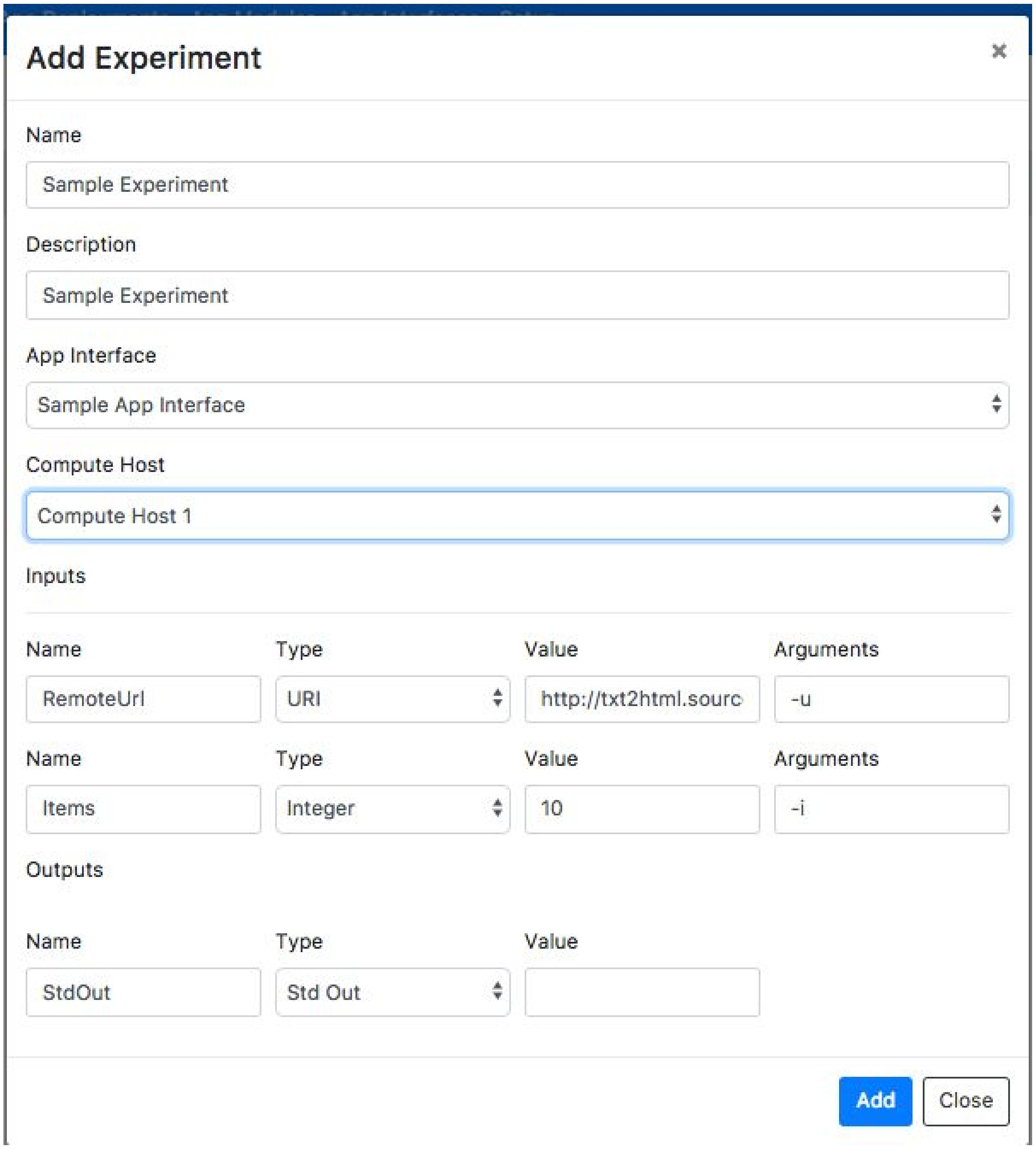
Inputs

Name	Type	Value	Arguments
RemoteUrl	URI	http://txt2html.sourc	-u
Items	Integer	10	-i

Outputs

Name	Type	Value
StdOut	Std Out	

7. Create an Experiment. Select previously created App Interface and Compute Resource.Inputs and outputs are fetched from the selected Application interface



Add Experiment

Name: Sample Experiment

Description: Sample Experiment

App Interface: Sample App Interface

Compute Host: Compute Host 1

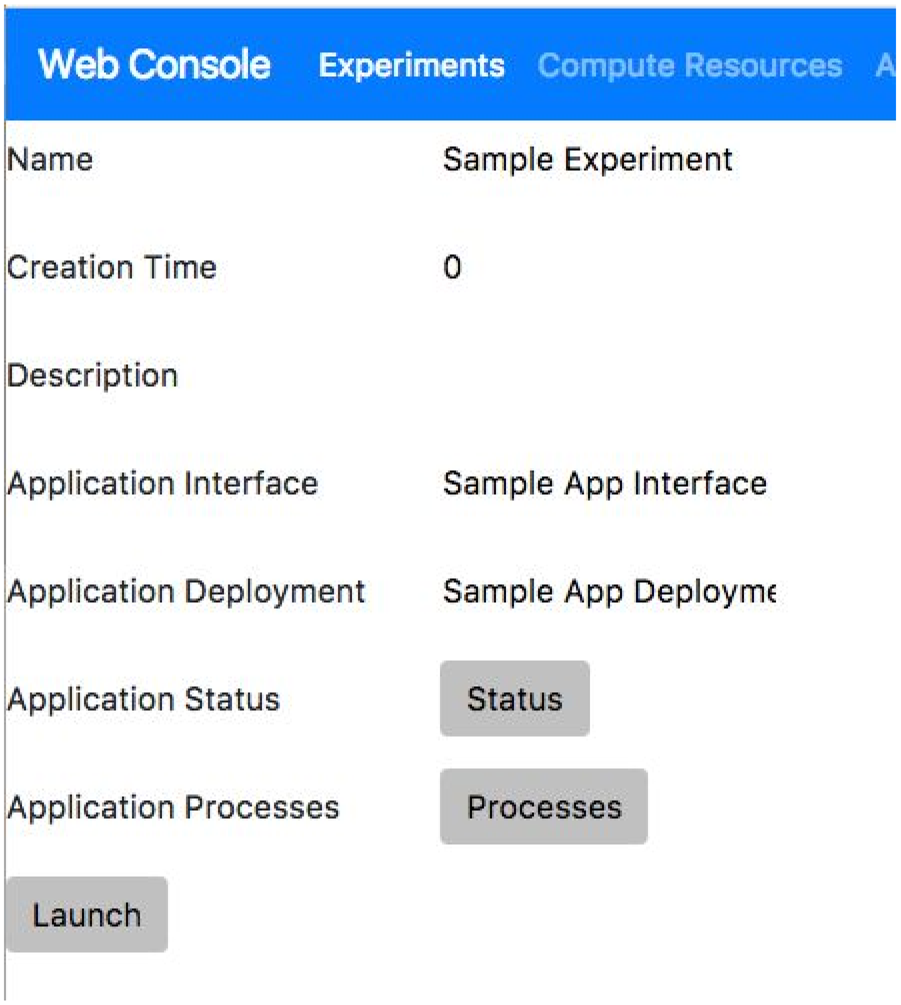
Inputs

Name	Type	Value	Arguments
RemoteUrl	URI	http://txt2html.sourc	-u
Items	Integer	10	-i

Outputs

Name	Type	Value
StdOut	Std Out	

8. To launch the experiment, go to Details page of the experiment and click Launch button.



Name Sample Experiment

Creation Time 0

Description

Application Interface Sample App Interface

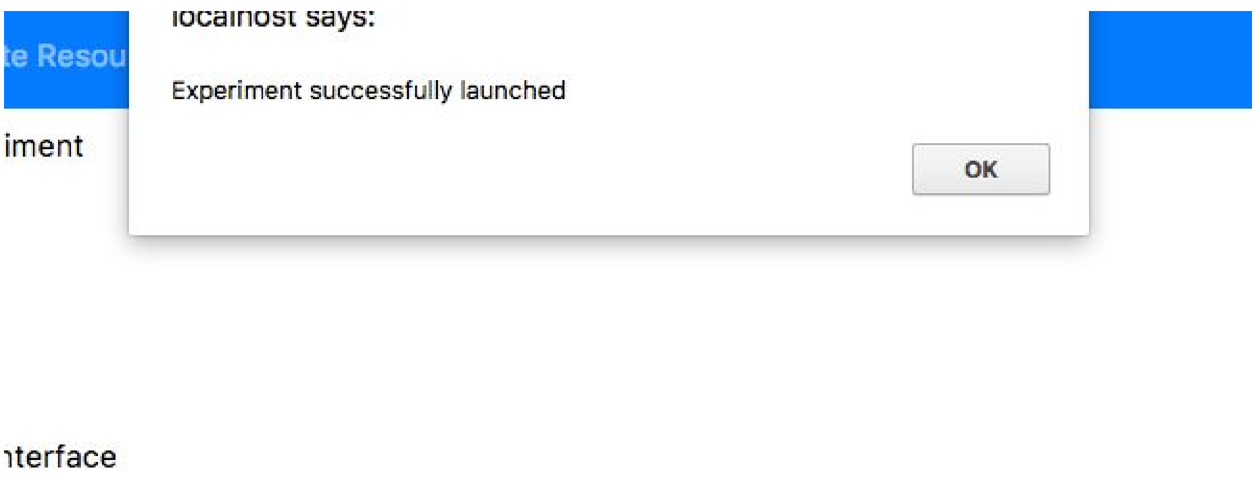
Application Deployment Sample App Deployme

Application Status Status

Application Processes Processes

Launch

If the experiment has successfully launched, it should show a message like this.

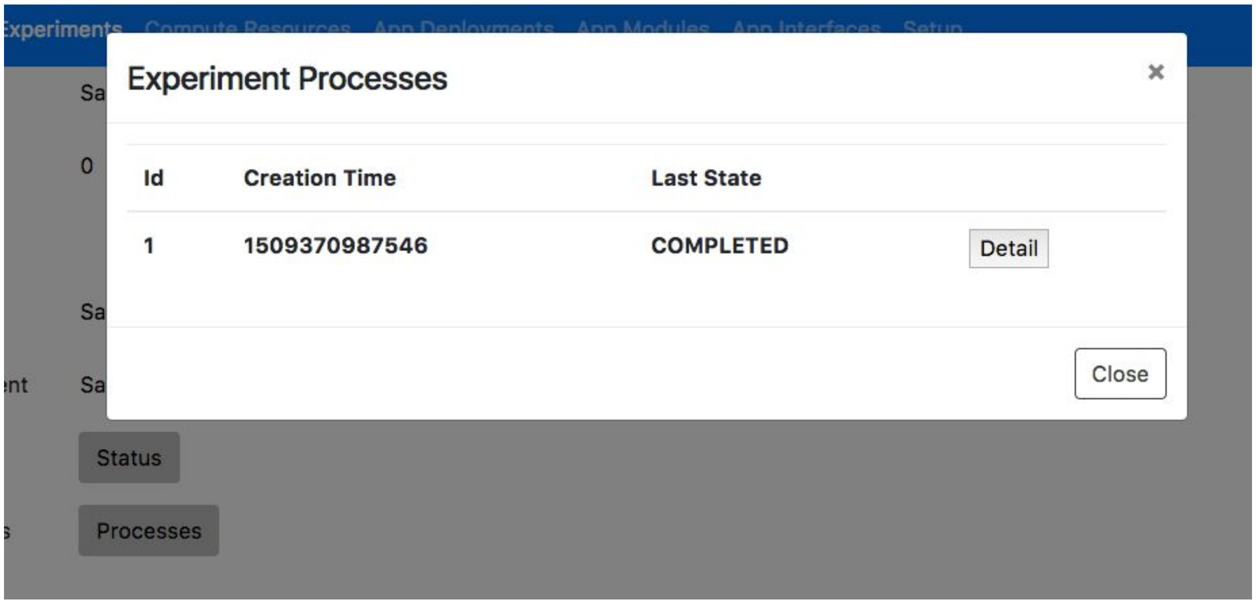


localhost says:

Experiment successfully launched

OK

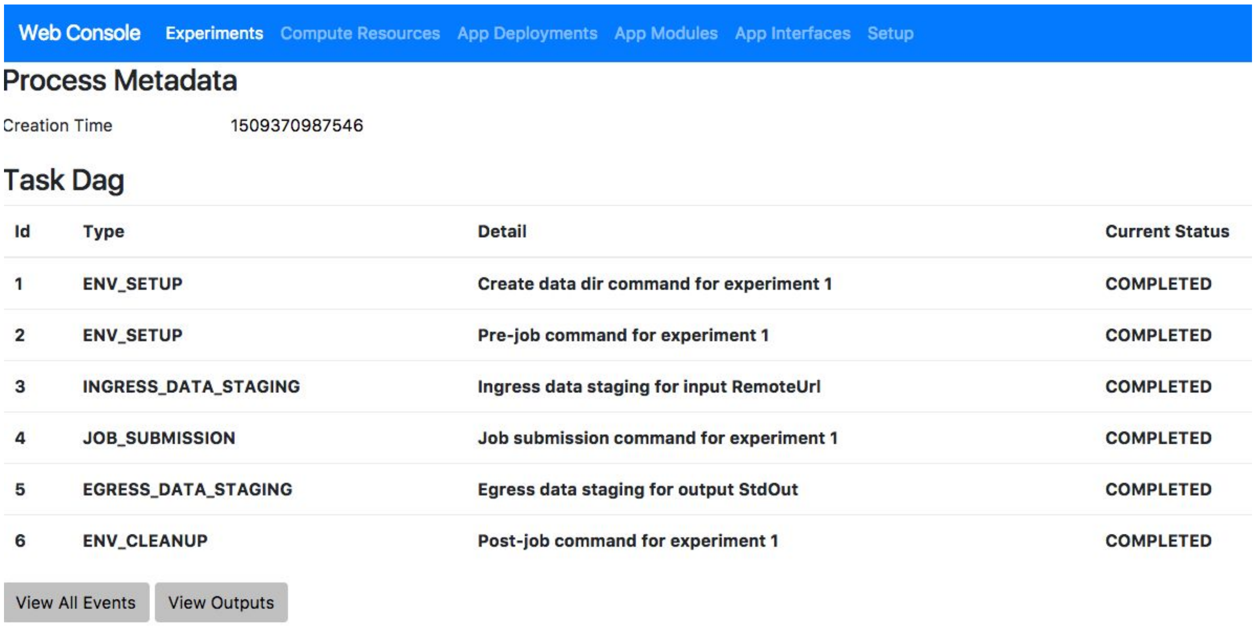
10. Click on the processes button and you can view currently running Processes under this Experiment



Experiment Processes

Id	Creation Time	Last State	
1	1509370987546	COMPLETED	Detail

11. If you click on the Details button, you can see the task dag generated for the process and their execution state. If all Tasks are Completed, Process is considered to be Completed



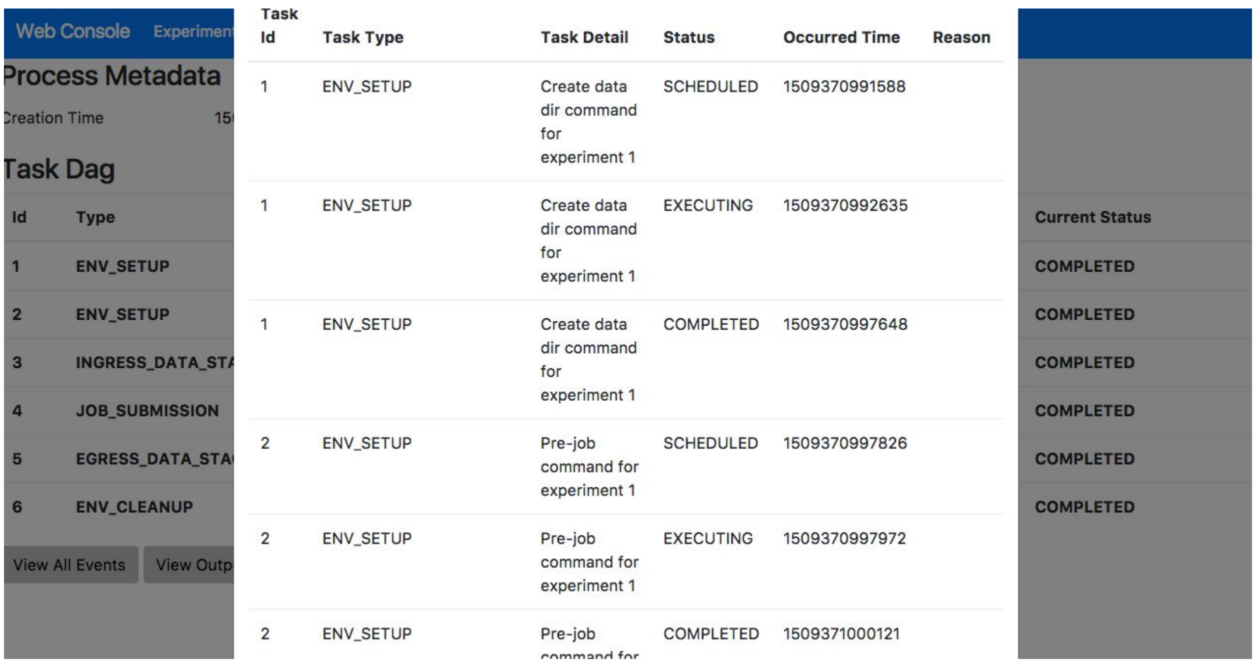
Process Metadata

Creation Time 1509370987546

Task Dag

Id	Type	Detail	Current Status
1	ENV_SETUP	Create data dir command for experiment 1	COMPLETED
2	ENV_SETUP	Pre-job command for experiment 1	COMPLETED
3	INGRESS_DATA_STAGING	Ingress data staging for input RemoteUrl	COMPLETED
4	JOB_SUBMISSION	Job submission command for experiment 1	COMPLETED
5	EGRESS_DATA_STAGING	Egress data staging for output StdOut	COMPLETED
6	ENV_CLEANUP	Post-job command for experiment 1	COMPLETED

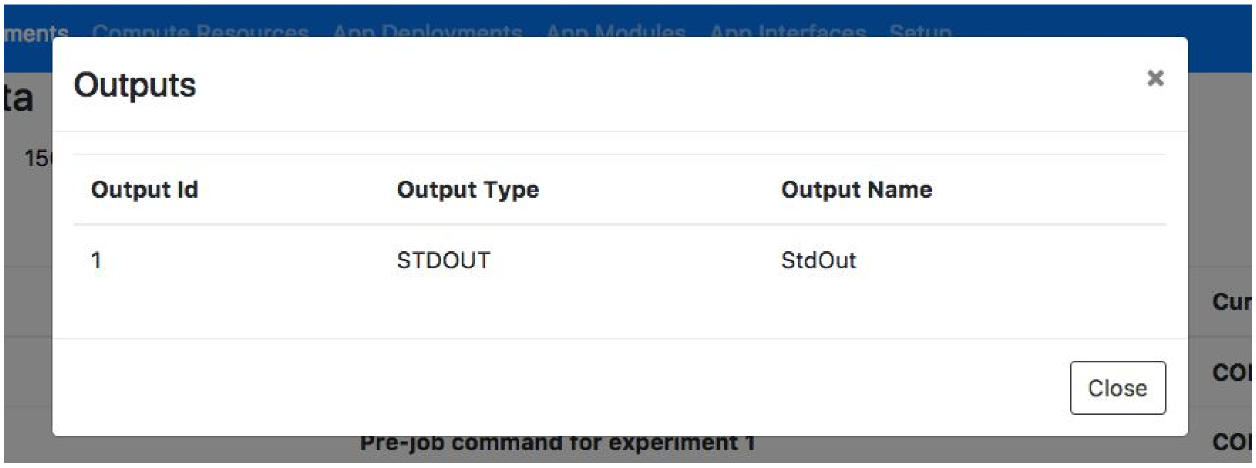
12. If you want to view the history of all the events occurred inside this process, click View All Events button.



Task Id	Task Type	Task Detail	Status	Occurred Time	Reason
1	ENV_SETUP	Create data dir command for experiment 1	SCHEDULED	1509370991588	
1	ENV_SETUP	Create data dir command for experiment 1	EXECUTING	1509370992635	
1	ENV_SETUP	Create data dir command for experiment 1	COMPLETED	1509370997648	
2	ENV_SETUP	Pre-job command for experiment 1	SCHEDULED	1509370997826	
2	ENV_SETUP	Pre-job command for experiment 1	EXECUTING	1509370997972	
2	ENV_SETUP	Pre-job command for	COMPLETED	1509371000121	

Id	Type	Current Status
1	ENV_SETUP	COMPLETED
2	ENV_SETUP	COMPLETED
3	INGRESS_DATA_STA	COMPLETED
4	JOB_SUBMISSION	COMPLETED
5	EGRESS_DATA_STA	COMPLETED
6	ENV_CLEANUP	COMPLETED

13. Currently fetched Output for the Process can be viewed by clicking View Outputs command



ments Compute Resources App Deployments App Modules App Interfaces Setup
Outputs
×
Close

Output Id	Output Type	Output Name
1	STDOUT	StdOut

Stop the platform

Once the testing is completed, shout down the platform by executing following command

```
minikube stop
```

In case if you want to start it again, simply run

```
minikube start
```