[YARN-4793] Simplified API layer for services and beyond

Table of Contents

1. Overview

Bringing a new service on YARN today is not a simple experience. The APIs of existing frameworks are either too low level (native YARN), require writing new code (for frameworks with programmatic APIs) or writing a complex spec (for declarative frameworks). In addition to building critical building blocks inside YARN (as part of other efforts at YARN-4692), there is a need for simplifying the user facing story for building services. Experience of projects like Apache Slider running real-life services like HBase, Storm, Accumulo, Solr etc, gives us some very good insights on how simplified APIs for services should look like.

To this end, we should look at a new simple-services API layer backed by REST interfaces. This API can be used to create and manage the lifecycle of YARN services. Services here can range from simple single-component apps to complex multi-component assemblies needing orchestration.

We should also look at making this a unified REST based entry point for other important features like resource-profile management (YARN-3926), package-definitions' lifecycle-management and service-discovery (YARN-913/YARN-4757). We also need to flesh out its relation to our present much lower level REST APIs (YARN-1695) in YARN for application-submission and management.

This document spotlights on this specification. In most of the cases, the application owner will not be forced to make any changes to their application. This is primarily true if the application is packaged with containerization technologies like docker. Irrespective of how complex the application is, there will be hooks provided at appropriate layers to allow pluggable and customizable application behavior.

1.1. Version information

Version: 1.0.0

1.2. License information

License: Apache 2.0 License URL: http://www.apache.org/licenses/LICENSE-2.0.html

1.3. URI scheme

Host: host.mycompany.com BasePath: /services/v1/ Schemes: HTTP, HTTPS

1.4. Consumes

- application/json

1.5. Produces

- application/json

2. Paths

2.1. List of applications/services running in the cluster

```
GET /applications
```

2.1.1. Description

Get a list of all currently running applications (response includes a minimal projection of the application info). For more details do a GET on a specific application name.

2.1.2. Responses

HTTP Code	Description	Schema
200	An array of applications	Application array
default	Unexpected error	Error

2.2. Create an application/service

```
POST /applications
```

2.2.1. Description

Create an application. The request JSON is an Application object with details required for creation. If the request is successful it returns 202 Accepted. A success of this API only confirms success in submission of the application creation request. There is no guarantee that the application will actually reach a RUNNING state. Resource availability and several other factors determines if the application will be deployed in the cluster. It is expected that clients would subsequently call the GET API to get details of the application and determine its state.

2.2.2. Parameters

Type	Name	Description	Required	Schema	Default
BodyParameter	Application	Application request object	true	Application	

2.2.3. Responses

HTTP Code	Description	Schema
202	Request accepted	No Content

HTTP Code	Description	Schema
default	Unexpected error	Error

2.3. Get an application/service details

```
GET /applications/{app_name}
```

2.3.1. Description

Return the details (including containers) of a running application

2.3.2. Parameters

Type	Name	Description	Required	Schema	Default
PathParameter	app_name	Application name	true	string	

2.3.3. Responses

HTTP Code	Description	Schema
200	An application object	object
404	Application does not exist	No Content
default	Unexpected error	Error

2.4. Update an application/service or upgrade the binary version of the components of a running application

```
PUT /applications/{app_name}
```

2.4.1. Description

Update the runtime properties of an application. As of now, only update of lifetime and number of instances (flexing) of the components of an application is supported. The PUT operation is also used to orchestrate an upgrade of the application containers to a newer version of their artifacts.

2.4.2. Parameters

Type	Name	Description	Required	Schema	Default
PathParameter	app_name	Application name	true	string	

2.4.3. Responses

HTTP Code	Description	Schema
204	Update or upgrade was successful	No Content
404	Application does not exist	No Content
default	Unexpected error	Error

2.5. Destroy application/service

```
DELETE /applications/{app_name}
```

2.5.1. Description

Destroy an application and release all resources. This API might have to return JSON data providing location of logs, etc. Not finalized yet.

2.5.2. Parameters

Type	Name	Description	Required	Schema	Default
PathParameter	app_name	Application name	true	string	

2.5.3. Responses

HTTP Code	Description	Schema
204	Destroy was successful	No Content
404	Application does not exist	No Content
default	Unexpected error	Error

3. Definitions

3.1. Application

An Application resource has the following attributes.

Name	Description	Required	Schema	Default
name	A unique application name	true	string	
artifact	Artifact of single-component applications. Mandatory if components attribute is not specified.	false	Artifact	
resource	Resource of single-component applications or the global default for multi-component applications. Mandatory if it is a single-component application and if cpus and memory are not specified at the Application level.	false	Resource	
launch_command	The custom launch command of an application component (optional). If not specified for applications with docker images say, it will default to the default start command of the image. If there is a single component in this application, you can specify this without the need to have a 'components' section.	false	string	
number_of_containers	Number of containers for each app-component in the application. Each app-component can further override this app-level global default.	false	integer (int64)	1
lifetime	Life time (with units) of the application from the time it reaches the RUNNING state (after which it is automatically destroyed by YARN). E.g. - 30mins, 10hours, 5days. For unlimited lifetime (long running) specify unlimited.	false	string	unlimited
placement_policy	Advanced scheduling and placement policies (optional). If not specified, it defaults to the default placement policy of the app owner. The design of placement policies are in the works. It is not very clear at this point, how policies in conjunction with labels be exposed to application owners. This is a placeholder for now. The advanced structure of this attribute will be determined by YARN-4902.	false	PlacementPolicy	
components	Components of an application	false	Component array	

Name	Description	Required	Schema	Default
configuration	Config properties of an application. Configurations provided at the application/global level are available to all the components. Specific properties can be overridden at the component level.	false	Configuration	
containers	Containers of a running application. Specifying a value for this attribute for the POST payload raises a validation error. This blob is available only in the GET response of a running application.	false	Container array	
state	State of the application. Specifying a value for this attribute for the POST payload raises a validation error. This attribute is available only in the GET response of a running application.	false	ApplicationState	

3.2. ApplicationState

The current state of an application

Name	Description	Required	Schema	Default
state	enum of the state of the application	false	enum (accepted, running, finished, failed, stopped, started)	

3.3. ApplicationStatus

The current status of a submitted application, returned as a response to the GET api

Name	Description	Required	Schema	Default
error_message	An error description (if any) for the reason of the current state of the application. It typically has a non-null value, if the application is in a non-running state.	false	string	
state	Application state	false	ApplicationState	

3.4. Artifact

Artifact of an application component

Name	Description	Required	Schema	Default
id	Artifact id. Examples are package location uri for tarball based apps, image name for docker, etc.	true	string	
type	Artifact type, like docker, tarball, etc. (optional)	false	enum (docker, tarball)	docker
uri	Artifact location to support multiple artifact stores (optional)	false	string	

3.5. Component

One or more components of the application. If the application is HBase say, then the component can be a simple role like master or regionserver. If the application is a complex business webapp then a component can be other applications say Kafka or Storm. Thereby it opens up the support for complex and nested applications.

Name	Description	Required	Schema	Default
name	Name of the application component (mandatory)	true	string	
dependencies	An array of application components which should be in READY state (as defined by readiness check), before this component can be started. The dependencies across all components of an application should be represented as a DAG.	false	string array	
readiness_check	Readiness check for this app-component	false	ReadinessCheck	
artifact	Artifact of the component (optional). If not specified, the application level global artifact takes effect.	false	Artifact	
launch_command	The custom launch command of this component (optional). When specified at the component level, it overrides the value specified at the global level (if any).	false	string	
resource	Resource of this component (optional). If not specified, the application level global resource takes effect.	false	Resource	

Name	Description	Required	Schema	Default
number_of_containers	Number of containers for this app-component (optional). If not specified, the application level global number_of_containers takes effect.	false	integer (int64)	1
unique_component_support	Certain applications need to define multiple components using the same artifact and resource profile, differing only in configurations. In such cases, this field helps app owners to avoid creating multiple component definitions with repeated information. The number_of_containers field dictates the initial number of components created. Component names typically differ with a trailing id, but assumptions should not be made on that, as the algorithm can change at any time. Configurations section will be able to use placeholders like ${USER}, ${CLUSTER_NAME} and ${COMPONENT_NAME} to be replaced at runtime with user the app is submitted as, application name and application component name respectively. Launch command can use placeholders like ${APP_COMPONENT_NAME} and ${APP_NAME} to get its component name and app name respectively at runtime. The best part of this feature is that when the component is flexed up, entirely new components (with new trailing ids) are created.	false	boolean	
run_privileged_container	Run all containers of this component in privileged mode (YARN-4262)	false	boolean	
placement_policy	Advanced scheduling and placement policies for all containers of this component (optional). If not specified, the app level placement_policy takes effect. Refer to the description at the global level for more details.	false	PlacementPolicy	
configuration	Config properties for this app-component	false	Configuration	

3.6. ConfigFile

A config file that needs to be created and made available as a volume in an application component container.

Name	Description	Required	Schema	Default
type	Config file in the standard format like xml, properties, json, yaml, template	false	enum (xml, properties, json, yaml, template, env, hadoop_xml)	
dest_file	The absolute path that this configuration file should be mounted as, in the application container	false	string	
src_file	Required for type template. This provides the source location of the template which needs to be mounted as dest_file post property substitutions. Typically the src_file would point to a source controlled network accessible file maintained by tools like puppet, chef, etc.	false	string	
props	A blob of key value pairs that will be dumped in the dest_file in the format as specified in type. If the type is template then the attribute src_file is mandatory and the src_file content is dumped to dest_file post property substitutions.	false	object	

3.7. Configuration

Set of configuration properties that can be injected into the application components via envs, files and custom pluggable helper docker containers. Files of several standard formats like xml, properties, json, yaml and templates will be supported.

Name	Description	Required	Schema	Default
properties	A blob of key-value pairs of common application properties	false	object	
env	A blob of key-value pairs which will be appended to the default system properties and handed off to the application at start time. All placeholder references to properties will be substituted before injection.	false	object	
files	Array of list of files that needs to be created and made available as volumes in the application component containers.	false	ConfigFile array	

3.8. Container

An instance of a running application container

Name	Description	Required	Schema	Default
id	Unique container id of a running application, e.g. container_e3751_1458061340047_0008_01_000002	false	string	
launch_time	The time when the container was created, e.g. 2016-03-16T01:01:49.000Z. This will most likely be different from cluster launch time.	false	string (date)	
ip	IP address of a running container, e.g. 172.31.42.141. The IP address and hostname attribute values are dependent on the cluster/docker network setup as per YARN-4007.	false	string	
hostname	Fully qualified hostname of a running container, e.g. ctr-e3751-1458061340047-0008-01-000002.examplestg.site. The IP address and hostname attribute values are dependent on the cluster/docker network setup as per YARN-4007.	false	string	
bare_host	The bare node or host in which the container is running, e.g. cn008.example.com	false	string	
state	State of the container of an application	false	ContainerState	
component_name	Name of the component that this container instance belongs to	false	string	
resource	Resource used for this container	false	Resource	

3.9. ContainerState

The current state of the container of an application

Name	Description	Required	Schema	Default
state	enum of the state of the container	false	enum (ready, init)	

3.10. Error

Name	Description	Required	Schema	Default
code		false	integer (int32)	
message		false	string	
fields		false	string	

3.11. PlacementPolicy

Placement policy of an instance of an application. This feature is in the works in YARN-4902.

Name	Description	Required	Schema	Default
label	Assigns an app to a named partition of the cluster where the application desires to run (optional). If not specified all apps are submitted to a default label of the app owner. One or more labels can be setup for each application owner account with required constraints like no-preemption, sla-99999, preemption-ok, etc.	false	string	

3.12. ReadinessCheck

A custom command or a pluggable helper container to determine the readiness of a container of a component. Readiness for every application is different. Hence the need for a simple interface, with scope to support advanced usecases.

Name	Description	Required	Schema	Default
type	http (YARN will perform a simple REST call at a regular interval and expect a 204 No content)	false	enum (http)	
uri	Fully qualified REST uri endpoint	true	string	
artifact	Artifact of the pluggable readiness check helper container (optional). If specified, this helper container typically hosts the http uri and encapsulates the complex scripts required to perform actual container readiness check. At the end it is expected to respond a 204 No content just like the simplified use case. This pluggable framework benefits application owners who can run applications without any packaging modifications. Note, artifacts of type docker only is supported for now.	false	Artifact	

3.13. Resource

Resource determines the amount of resources (vcores, memory, network, etc.) usable by a container. This field determines the resource to be applied for all the containers of a component or application. The resource specified at the app (or global) level can be overriden at the component level. Only one of profile OR cpu & memory are exepected. It raises a validation exception otherwise.

Name	Description	Required	Schema	Default
profile	Each resource profile has a unique id which is associated with a cluster-level predefined memory, cpus, etc.	false	string	
cpus	Amount of vcores allocated to each container (optional but overrides cpus in profile if specified)	false	integer (int32)	
memory	Amount of memory allocated to each container (optional but overrides memory in profile if specified). Currently accepts only an integer value and default unit is in MB.	false	string	

4. Examples

4.1. Create a simple single-artifact application with most attribute values as defaults

POST URL - http://host.mycompany.com:8088/services/v1/applications/

4.1.1. POST Request JSON

```
{
  "name": "hello-world",
  "artifact": {
    "id": "nginx:latest"
  },
  "resource": {
    "cpus": 1,
    "memory": "2048"
  }
}
```

4.1.2. GET Response JSON

GET URL - http://host.mycompany.com:8088/services/v1/applications/hello-world

```
{
  "uri": "/services/v1/applications/hello-world",
  "name": "hello-world",
  "lifetime": "unlimited",
  "state": "RUNNING",
  "number_of_containers": 1,
  "containers": [
    {
      "uri": "/services/v1/applications/hello-
world/containers/container_e3751_1458061340047_0008_01_000002",
      "id": "container_e3751_1458061340047_0008_01_000002",
      "launch_time": "2016-03-16T01:01:49.000Z",
      "ip": "172.31.42.141",
      "hostname": "ctr-e3751-1458061340047-0008-01-000002.examplestg.site",
      "state": "READY",
      "component_name": "DEFAULT",
      "bare_host": "cn007.example.com",
      "resource": {
        "cpus": 1,
        "memory": "2048"
      }
    }
  ]
}
```


4.2. Create a simple single-artifact application

POST URL - http://host.mycompany.com:8088/services/v1/applications/

Showing overrides of default attribute values

4.2.1. POST Request JSON

```
{
  "name": "hello-world",
  "lifetime": "100hours",
  "number_of_containers": 3,
  "artifact": {
    "id": "nginx:latest",
    "type": "DOCKER"
  },
  "launch_command": "/start_nginx.sh",
  "readiness_check": {
    "uri": "${APP_NAME}-${APP_COMPONENT_NAME}-${APP_COMPONENT_ID}.examplestg.site",
    "artifact": {
      "id": "readiness-check/nginx:latest",
      "type": "DOCKER"
    }
  },
  "resource": {
    "cpus": 2,
    "memory": "16384"
  }
}
```

4.2.2. GET Response JSON

GET URL - http://host.mycompany.com:8088/services/v1/applications/hello-world-2

```
{
  "uri": "/services/v1/applications/hello-world-2",
  "name": "hello-world-2",
  "lifetime": "100hours",
  "state": "RUNNING",
  "number_of_containers": 3,
  "containers": [
    {
      "uri": "/services/v1/applications/hello-world-
2/containers/container_e3751_1458061340048_0008_01_000002",
      "id": "container_e3751_1458061340048_0008_01_000002",
      "launch_time": "2016-03-16T01:01:49.000Z",
      "ip": "172.31.42.142",
      "hostname": "ctr-e3751-1458061340048-0008-01-000002.examplestg.site",
      "state": "READY",
      "component_name": "DEFAULT",
      "bare_host": "cn007.example.com",
      "resource": {
        "cpus": 2,
        "memory": "16384"
      }
    },
    {
```

```
      "uri": "/services/v1/applications/hello-world-
2/containers/container_e3751_1458061340048_0008_01_000003",
      "id": "container_e3751_1458061340048_0008_01_000003",
      "launch_time": "2016-03-16T01:01:49.000Z",
      "ip": "172.31.42.143",
      "hostname": "ctr-e3751-1458061340048-0008-01-000003.examplestg.site",
      "state": "READY",
      "component_name": "DEFAULT",
      "bare_host": "cn008.example.com",
      "resource": {
        "cpus": 2,
        "memory": "16384"
      }
    },
    {
      "uri": "/services/v1/applications/hello-world-
2/containers/container_e3751_1458061340048_0008_01_000004",
      "id": "container_e3751_1458061340048_0008_01_000004",
      "launch_time": "2016-03-16T01:01:49.000Z",
      "ip": "172.31.42.144",
      "hostname": "ctr-e3751-1458061340048-0008-01-000004.examplestg.site",
      "state": "READY",
      "component_name": "DEFAULT",
      "bare_host": "cn009.example.com",
      "resource": {
        "cpus": 2,
        "memory": "16384"
      }
    }
  ]
}
```


4.3. Create HBase

POST URL - http://host.mycompany.com:8088/services/v1/applications/

4.3.1. POST Request JSON


```
{
  "name": "hbase-dept1",
  "lifetime": "200hours",
  "configurations": {
    "properties": {
      "app_version": "${hbase.version}",
      "app_root": "${AGENT_WORK_ROOT}/app/install/hbase-${hbase.version}",
      "metric_collector_host": "${NN_HOST}",
      "metric_collector_port": "6188",
      "metric_collector_lib": "",
      "hbase_instance_name": "instancename",
```

```
      "hbase_root_password": "secret",
      "user_group": "hadoop",
      "monitor_protocol": "http",
      "hbase_thrift_port": "${HBASE_THRIFT.ALLOCATED_PORT}",
      "hbase_thrift2_port": "${HBASE_THRIFT2.ALLOCATED_PORT}",
      "hbase_rest_port": "${HBASE_REST.ALLOCATED_PORT}"
    },
    "env": {
      "hbase_master_heapsize": "1024m",
      "hbase_regionserver_heapsize": "1024m"
    },
    "files": [
      {
        "type": "HADOOP_XML",
        "dest_file": "/etc/hadoop/conf/hbase-site.xml",
        "props": {
          "hbase.rootdir": "${DEFAULT_DATA_DIR}/data",
          "hbase.superuser": "${USER_NAME}",
          "hbase.tmp.dir": "${AGENT_WORK_ROOT}/work/app/tmp",
          "hbase.local.dir": "${hbase.tmp.dir}/local",
          "hbase.zookeeper.quorum": "${ZK_HOST}",
          "zookeeper.znode.parent": "${DEFAULT_ZK_PATH}",
          "hbase.regionserver.info.port": "0",
          "hbase.bulkload.staging.dir": "/user/${USER_NAME}/hbase-staging",
          "hbase.coprocessor.region.classes":
"org.apache.hadoop.hbase.security.access.SecureBulkLoadEndpoint",
          "hbase.master.info.port": "${HBASE_MASTER.ALLOCATED_PORT}",
          "hbase.regionserver.port": "0",
          "hbase.master.port": "0"
        }
      }
    ]
  },
  "components": [
    {
      "name": "HBASE_MASTER",
      "dependencies": [  ],
      "readiness_check": {
        "type": "HTTP",
        "uri": "hmaster.${APP_NAME}-${APP_COMPONENT_NAME}.root.examplestg.site",
        "artifact": {
          "id": "readiness-check/hbase-master:latest",
          "type": "DOCKER"
        }
      },
      "number_of_containers": 1,
      "artifact": {
        "id": "hbase:latest",
        "type": "DOCKER"
      },
      "launch_command": "/hbase_master.py",
```

```
      "resource": {
        "profile": "xlarge"
      }
    },
    {
      "name": "HBASE_REGIONSERVER",
      "dependencies": [ "hbase_master" ],
      "readiness_check": {
        "type": "HTTP",
        "uri": "hregion${APP_COMPONENT_ID}.${APP_NAME}-
${APP_COMPONENT_NAME}.root.examplestg.site",
        "artifact": {
          "id": "readiness-check/hbase-region:latest",
          "type": "DOCKER"
        }
      },
      "number_of_containers": 2,
      "artifact": {
        "id": "hbase:latest",
        "type": "DOCKER"
      },
      "launch_command": "/hbase_region.py",
      "resource": {
        "profile": "medium"
      },
      "placement_policy": {
        "label": "anti_affinity"
      }
    },
    {
      "name": "HBASE_DATA_MONITOR",
      "dependencies": [  ],
      "readiness_check": {
        "type": "HTTP",
        "uri": "hbase_mon.${APP_NAME}-${APP_COMPONENT_NAME}.root.examplestg.site",
        "artifact": {
          "id": "readiness-check/hbase-mon:latest",
          "type": "DOCKER"
        }
      },
      "number_of_containers": 1,
      "artifact": {
        "id": "hbase-data-monitor:latest",
        "type": "DOCKER"
      },
      "launch_command": "/hbase_data_monitor_start.sh > /tmp/hbase_data_monitor.log
2>&1",
      "configurations": {
        "env": {
          "DATA_DIR": "${DEFAULT_DATA_DIR}"
        }
```

```
      },
      "resource": {
        "cpus": 1,
        "memory": "4096"
      }
    }
  ]
}
```

4.3.2. GET Response JSON

GET URL - http://host.mycompany.com:8088/services/v1/applications/hbase-dept1

```
{
  "uri": "/services/v1/applications/hbase-dept1",
  "name": "hbase-dept1",
  "lifetime": "200hours",
  "state": "RUNNING",
  "number_of_containers": 4,
  "containers": [
    {
      "uri": "/services/v1/applications/hbase-
dept1/containers/container_e3751_1458061340049_0008_01_000002",
      "id": "container_e3751_1458061340049_0008_01_000002",
      "launch_time": "2016-03-16T01:01:49.000Z",
      "ip": "172.31.42.145",
      "hostname": "ctr-e3751-1458061340049-0008-01-000002.examplestg.site",
      "state": "READY",
      "component_name": "HBASE_MASTER",
      "bare_host": "cn007.example.com",
      "resource": {
        "cpus": 4,
        "memory": "16384"
      }
    },
    {
      "uri": "/services/v1/applications/hbase-
dept1/containers/container_e3751_1458061340049_0008_01_000003",
      "id": "container_e3751_1458061340049_0008_01_000003",
      "launch_time": "2016-03-16T01:01:49.000Z",
      "ip": "172.31.42.146",
      "hostname": "ctr-e3751-1458061340049-0008-01-000003.examplestg.site",
      "state": "READY",
      "component_name": "HBASE_REGIONSERVER",
      "bare_host": "cn008.example.com",
      "resource": {
        "cpus": 2,
        "memory": "8192"
      }
    },
```

```
    {
      "uri": "/services/v1/applications/hbase-
dept1/containers/container_e3751_1458061340049_0008_01_000004",
      "id": "container_e3751_1458061340049_0008_01_000004",
      "launch_time": "2016-03-16T01:01:49.000Z",
      "ip": "172.31.42.147",
      "hostname": "ctr-e3751-1458061340049-0008-01-000004.examplestg.site",
      "state": "READY",
      "component_name": "HBASE_REGIONSERVER",
      "bare_host": "cn009.example.com",
      "resource": {
        "cpus": 2,
        "memory": "8192"
      }
    },
    {
      "uri": "/services/v1/applications/hbase-
dept1/containers/container_e3751_1458061340049_0008_01_000005",
      "id": "container_e3751_1458061340049_0008_01_000005",
      "launch_time": "2016-03-16T01:01:49.000Z",
      "ip": "172.31.42.148",
      "hostname": "ctr-e3751-1458061340049-0008-01-000005.examplestg.site",
      "state": "READY",
      "component_name": "HBASE_DATA_MONITOR",
      "bare_host": "cn005.example.com",
      "resource": {
        "cpus": 1,
        "memory": "4096"
      }
    }
  ]
}
```


4.4. Create a complex business application with multiple components

POST URL - http://host.mycompany.com:8088/services/v1/applications/

Showing override of attribute values at the global level by values at the component level

4.4.1. POST Request JSON


```
{
  "name": "logsearch4",
  "lifetime": "unlimited",
  "resource": {
    "cpus": 2,
    "memory": "8192"
```

```
  },
  "components": [
    {
      "name": "ZK",
      "dependencies": [  ],
      "readiness_check": {
        "type": "HTTP",
        "uri": "zk${APP_COMPONENT_ID}.${APP_NAME}-
${APP_COMPONENT_NAME}.root.examplestg.site",
        "artifact": {
          "id": "readiness-check/zookeeper:latest",
          "type": "DOCKER"
        }
      },
      "number_of_containers": 3,
      "artifact": {
        "id": "zookeeper:latest"
      },
      "unique_component_support": "true",
      "launch_command": "/wait_for_hosts.sh zk1.${APP_NAME}-
${APP_COMPONENT_NAME}.root.examplestg.site zk2.${APP_NAME}-
${APP_COMPONENT_NAME}.root.examplestg.site zk3.${APP_NAME}-
${APP_COMPONENT_NAME}.root.examplestg.site; JVMFLAGS=-Djava.net.preferIPv4Stack=true
/opt/zookeeper/bin/zkServer.sh start-foreground > /tmp/zoo.log 2>/tmp/zoo.err",
      "configurations": {
        "properties": {
          "dataDir":
"/grid/0/hadoop/yarn/local/usercache/${USER}/${CLUSTER_NAME}/${COMPONENT_NAME}"
        },
        "files": [
          {
            "type": "PROPERTIES",
            "dest_file": "${dataDir}/myid",
            "props": {
              "content": "${COMPONENT_ID}"
            }
          },
          {
            "type": "PROPERTIES",
            "dest_file": "/opt/zookeeper/conf/zoo.cfg",
            "props": {
              "tickTime": "2000",
              "initLimit": "10",
              "syncLimit": "5",
              "dataDir": "${dataDir}",
              "clientPort": "2181",
              "maxClientCnxns": "100",
              "server.1": "zk1.${CLUSTER_NAME}.${USER}.examplestg.site:2888:3888",
              "server.2": "zk2.${CLUSTER_NAME}.${USER}.examplestg.site:2888:3888",
              "server.3": "zk3.${CLUSTER_NAME}.${USER}.examplestg.site:2888:3888"
            }
```

```
          }
        ]
      },
      "resource": {
        "cpus": 2,
        "memory": "8096"
      },
      "placement_policy": {
        "label": "anti_affinity"
      }
    },
    {
      "name": "SOLR",
      "dependencies": [ "ZK" ],
      "readiness_check": {
        "type": "HTTP",
        "uri": "solr${APP_COMPONENT_ID}.${APP_NAME}-
${APP_COMPONENT_NAME}.root.examplestg.site",
        "artifact": {
          "id": "readiness-check/solr:latest",
          "type": "DOCKER"
        }
      },
      "number_of_containers": 2,
      "artifact": {
        "id": "solr:latest",
        "type": "DOCKER"
      },
      "unique_component_support": "true",
      "launch_command": "/solr_start_hdfs.sh -cloud -f -p $SOLR_PORT -m $SOLR_MEMORY
-z ${ZK_HOST}$ZK_PATH -Dsolr.directoryFactory=HdfsDirectoryFactory
-Dsolr.lock.type=none -Dsolr.hdfs.confdir=/etc/hadoop/conf -Dsolr.hdfs.home=$DATA_DIR
-Djava.net.preferIPv4Stack=true > /tmp/solr.log 2>&1",
      "configurations": {
        "properties": {
          "create.default.zookeeper.node": "true"
        },
        "env": {
          "SOLR_PORT": "8983",
          "SOLR_MEMORY": "8192m",
          "ZK_HOST": "zk1.${APP_NAME}-zk.root.examplestg.site,zk2.${APP_NAME}-
zk.root.examplestg.site,zk3.${APP_NAME}-zk.root.examplestg.site",
          "ZK_PATH": "/solr-hdfs",
          "DATA_DIR": "${DEFAULT_DATA_DIR}"
        }
      },
      "resource": {
        "cpus": 2,
        "memory": "8192"
      },
      "placement_policy": {
```

```
        "label": "strict"
      }
    },
    {
      "name": "UI",
      "dependencies": [ "SOLR" ],
      "readiness_check": {
        "type": "HTTP",
        "uri": "ui.${APP_NAME}-${APP_COMPONENT_NAME}.root.examplestg.site",
        "artifact": {
          "id": "readiness-check/ui:latest",
          "type": "DOCKER"
        }
      },
      "number_of_containers": 1,
      "artifact": {
        "id": "logsearch-portal:latest"
      },
      "launch_command": "/init_and_run_hdfs.sh",
      "configurations": {
        "properties": {
          "logsearch-portal-site.destDir": "/logsearch-portal/classes"
        },
        "env": {
          "NUM_SHARDS": "2",
          "ZK_HOST": "zk1.${APP_NAME}-zk.root.examplestg.site,zk2.${APP_NAME}-
zk.root.examplestg.site,zk3.${APP_NAME}-zk.root.examplestg.site",
          "ZK_PATH": "/solr-hdfs",
          "SOLR_SERVER": "solr1.${APP_NAME}-solr.root.examplestg.site",
          "SOLR_PORT": "8983",
          "PID_FILE": "/logsearch-portal/logsearch-portal.pid",
          "LOGFILE": "/logsearch-portal/logsearch-portal.log",
          "LOGSEARCH_JAVA_MEM": "-Xmx512M",
          "LOGSEARCH_JAVA_OPTS": "-Djava.net.preferIPv4Stack=true"
        },
        "files": [
          {
            "type": "PROPERTIES",
            "dest_file": "<path>/system.properties",
            "props": {
              "solr.zkhosts": "{ZK_HOST}{ZK_PATH}",
              "auditlog.solr.zkhosts": "{ZK_HOST}{ZK_PATH}"
            }
          }
        ]
      },
      "resource": {
        "profile": "small"
      }
    },
    {
```

```
      "name": "SOLR_DATA_MONITOR",
      "dependencies": [ "SOLR" ],
      "readiness_check": {
        "type": "HTTP",
        "uri": "solr_mon.${APP_NAME}-${APP_COMPONENT_NAME}.root.examplestg.site",
        "artifact": {
          "id": "readiness-check/data-mon:latest",
          "type": "DOCKER"
        }
      },
      "number_of_containers": 1,
      "artifact": {
        "id": "solr-data-monitor:latest"
      },
      "launch_command": "/solr_data_monitor_start.sh > /tmp/solr_data_monitor.log
2>&1",
      "configurations": {
        "env": {
          "DATA_DIR": "${DEFAULT_DATA_DIR}"
        }
      },
      "resource": {
        "cpus": 2,
        "memory": "4096"
      }
    }
  ]
}
```

4.4.2. GET Response JSON

GET URL - http://host.mycompany.com:8088/services/v1/applications/logsearch4

```
{
  "uri": "/services/v1/applications/logsearch4",
  "name": "logsearch4",
  "lifetime": "unlimited",
  "state": "RUNNING",
  "number_of_containers": 7,
  "containers": [
    {
      "uri":
"/services/v1/applications/logsearch4/containers/container_e3751_1458061340050_0008_01
_000002",
      "id": "container_e3751_1458061340050_0008_01_000002",
      "launch_time": "2016-03-16T01:01:49.000Z",
      "ip": "172.31.42.149",
      "hostname": "ctr-e3751-1458061340050-0008-01-000002.examplestg.site",
      "state": "READY",
      "component_name": "ZK",
```

```
      "bare_host": "cn008.example.com",
      "resource": {
        "cpus": 2,
        "memory": "8192"
      }
    },
    {
      "uri":
"/services/v1/applications/logsearch4/containers/container_e3751_1458061340050_0008_01
_000003",
      "id": "container_e3751_1458061340050_0008_01_000003",
      "launch_time": "2016-03-16T01:01:49.000Z",
      "ip": "172.31.42.150",
      "hostname": "ctr-e3751-1458061340050-0008-01-000003.examplestg.site",
      "state": "READY",
      "component_name": "ZK",
      "bare_host": "cn005.example.com",
      "resource": {
        "cpus": 2,
        "memory": "8192"
      }
    },
    {
      "uri":
"/services/v1/applications/logsearch4/containers/container_e3751_1458061340050_0008_01
_000004",
      "id": "container_e3751_1458061340050_0008_01_000004",
      "launch_time": "2016-03-16T01:01:49.000Z",
      "ip": "172.31.42.151",
      "hostname": "ctr-e3751-1458061340050-0008-01-000004.examplestg.site",
      "state": "READY",
      "component_name": "ZK",
      "bare_host": "cn007.example.com",
      "resource": {
        "cpus": 2,
        "memory": "8192"
      }
    },
    {
      "uri":
"/services/v1/applications/logsearch4/containers/container_e3751_1458061340050_0008_01
_000005",
      "id": "container_e3751_1458061340050_0008_01_000005",
      "launch_time": "2016-03-16T01:01:49.000Z",
      "ip": "172.31.42.152",
      "hostname": "ctr-e3751-1458061340050-0008-01-000005.examplestg.site",
      "state": "READY",
      "component_name": "SOLR",
      "bare_host": "cn008.example.com",
      "resource": {
        "cpus": 2,
```

```
        "memory": "8192"
      }
    },
    {
      "uri":
"/services/v1/applications/logsearch4/containers/container_e3751_1458061340050_0008_01
_000006",
      "id": "container_e3751_1458061340050_0008_01_000006",
      "launch_time": "2016-03-16T01:01:49.000Z",
      "ip": "172.31.42.153",
      "hostname": "ctr-e3751-1458061340050-0008-01-000006.examplestg.site",
      "state": "READY",
      "component_name": "SOLR",
      "bare_host": "cn005.example.com",
      "resource": {
        "cpus": 2,
        "memory": "8192"
      }
    },
    {
      "uri":
"/services/v1/applications/logsearch4/containers/container_e3751_1458061340050_0008_01
_000007",
      "id": "container_e3751_1458061340050_0008_01_000007",
      "launch_time": "2016-03-16T01:01:49.000Z",
      "ip": "172.31.42.154",
      "hostname": "ctr-e3751-1458061340050-0008-01-000007.examplestg.site",
      "state": "READY",
      "component_name": "UI",
      "bare_host": "cn007.example.com",
      "resource": {
        "cpus": 1,
        "memory": "4096"
      }
    },
    {
      "uri":
"/services/v1/applications/logsearch4/containers/container_e3751_1458061340050_0008_01
_000008",
      "id": "container_e3751_1458061340050_0008_01_000008",
      "launch_time": "2016-03-16T01:01:49.000Z",
      "ip": "172.31.42.155",
      "hostname": "ctr-e3751-1458061340050-0008-01-000008.examplestg.site",
      "state": "READY",
      "component_name": "SOLR_DATA_MONITOR",
      "bare_host": "cn009.example.com",
      "resource": {
        "cpus": 2,
        "memory": "4096"
      }
    }
```

```
  ]
}
```

4.5. Update to increase the lifetime of an application

PUT URL - http://host.mycompany.com:8088/services/v1/applications/hbase-dept1

4.5.1. PUT Request JSON

```
{
  "additional_lifetime": "24hours"
}
```

4.6. Update to flex up/down the no of containers (instances) of components of an application

PUT URL - http://host.mycompany.com:8088/services/v1/applications/hbase-dept1

4.6.1. PUT Request JSON

```
{
  "components": [
    {
      "name": "HBASE_REGIONSERVER",
      "number_of_containers": "+5"
    },
    {
      "name": "HBASE_REST",
      "number_of_containers": "-2"
    }
  ]
}
```

4.7. Upgrade a specific container of a component of an application to a newer version of an artifact

PUT URL - http://host.mycompany.com:8088/services/v1/applications/hbase-dept1

Orchestrate a rolling upgrade of an application. One or more currently running containers of a specific component of the application can be upgraded to a newer version. This can be done at once or with sufficient delays between each container upgrade. The API service does not provide any orchestration. It is the responsibility of the application owner to run a series of such PUT requests orchestrating the expected order of the upgrade and to ensure that the application as a whole is up and available to end users during the entire process. The id attribute needs to be provided for each

container in the PUT request. TODO: Can a different size of the running container be requested during upgrade?

4.7.1. PUT Request JSON

```
{
  "components": [
    {
      "id": "container_e3751_1458061340049_0008_01_000003",
      "name": "HBASE_REGIONSERVER",
      "readiness_check": {
        "type": "HTTP",
        "uri": "hregion${APP_COMPONENT_ID}.${APP_NAME}-
${APP_COMPONENT_NAME}.root.examplestg.site",
        "artifact": {
          "id": "readiness-check/hbase-region:latest",
          "type": "DOCKER"
        }
      },
      "artifact": {
        "id": "hbase:version20160601",
        "type": "DOCKER"
      },
      "launch_command": "/hbase_region_new.py",
      "resource": {
        "cpus": 4,
        "memory": "4096"
      }
    }
  ]
}
```

4.8. Destroy an application

DELETE URL - http://host.mycompany.com:8088/services/v1/applications/hbase-dept1