Servitiom

SARE SA

Oct 31, 2018

Contents

CHAPTER 1

Services

Service entity in **Servitiom** represents microservice application, who can be used by registered tenants. It can be deployed in many instances with unique custom properties like replicas count, resources limits.

1.1 Service Instance

Service instance in Servitiom represents unique deployed service in specified version and assigned to tenant. One tenant can have only one instance of service.

1.2 Versioning

Service can have multiply registered versions. Each version contains jobs templates for instance and service contexts. Template Schema:

```
schemaVersion: "0.1"
serviceVersion: "1.0.0"
jobs:
    instance:
        deploy:
```

1.3 ServiceTask

ServiceTask defining how execute some work in service like deploy instance. Entity fields:

- id - unique id in service
- name - human friendly name of task
- description - some text to describe task purpose

- scope: instance | service
- versions: * default - special version to use when not explicity in request * <version_id>
  - template:

```
id: <string>
```

Deploying service instance can consist of many steps, with simple template you can describe what must be done to run instance of your service. For more info about templating check: https://twig.symfony.com/

POST: https://<servitiom_api_uri>/v1/services id: "test" name: "test service", description: "simple service description bla bla bla"

PUT: https://<servitiom_api_uri>/v1/services/test/variables/api_password <value>

PUT: https://<servitiom_api_uri>/v1/variables/DOCKER_REGISTRY <value>

PUT: https://<servitiom_api_uri>/v1/services/test/versions/<version>/instance

{{set serviceAccountApiUri = "https://api.example.com/test/v1/service/accounts/#{InstanceInfo.customs.accountId}"}} schemaVersion: 1.0 x-moduleCommon: &module_common

name: "{{ServiceInfo.id}}_{{InstanceInfo.customs.accountId}}"

steps:

onDeploy:

RegisterInApi: type: ApiCall applyCondition: {{InstanceInfo.customs.accountId == 100}} parameters:

method: PUT uri: "{{serviceAccountApiUri}}" options:

auth: ["username", "{{ServiceInfo.variables.api_password}}"]

DeployModule: type: DockerServiceCreate parameters:

<<: *module_common image: "{{Variables.DOCKER_REGISTRY}}_{{ServiceInfo.id}}_module:1.0.0" env:

ACCOUNTID: "{{instanceInfo.customs.accountId}}"

onRemove:

UnregisterInApi: type: ApiCall parameters:

method: DELETE uri: "{{serviceAccountApiUri}}"

RemoveModule: type: DockerServiceRemove parameters:

name: "{{moduleServiceName}}"

POST: https://<servitiom_api_uri>/v1/services/<serviceId>/instances customs:

accountId: 1 extra2: "test" environment: "PROD"

1.3.1 Schema

```
apiVersion: v1
name: # human friendly name of service
steps:
  upgrade_from: # represents list of steps to upgrade from specify version of service
```

(continues on next page)

(continued from previous page)

```
    <version>:
      steps:
        <stepid>: # unique id of step
          type: # type of step must be one of supported step types like ApiCall
          name: # human friendly name of step, optional
          description: # some text to describe step operations, optional
          parameters: # some step depends parameters
    default:
  deploy: # represents list of steps to deploy new service instance
    <stepid>: # unique id of step
      type: # type of step must be one of supported step types like ApiCall
      name: # human friendly name of step, optional
      description: # some text to describe step operations, optional
      parameters: # some step depends parameters
  remove:  # represents list of steps to remove service instance
    <stepid>: # unique id of step
      type: # type of step must be one of supported step types like ApiCall
      name: # human friendly name of step, optional
      description: # some text to describe step operations, optional
      parameters: # some step depends parameters
```

Example

```
{{set moduleServiceName = "#{serviceInfo.name}_#{instanceInfo.customs.accountId}"}}
{{set serviceAccountApiUri = "#{serviceInfo.subservices.api.baseUri}/service/accounts/
↪#{instanceInfo.customs.accountId}"}}
apiVersion: v1
steps:
  deploy:
    RegisterInApi:
      type: ApiCall
      parameters:
        method: PUT
        uri: "{{serviceAccountApiUri}}"
    DeployModule:
      type: DockerServiceCreate
      parameters:
      name: "{{moduleServiceName}}"
      image: "{{serviceInfo.subservices.module.image}}"
      env:
        - ACCOUNTID="{{instanceInfo.customs.accountId}}"
  remove:
    UnregisterInApi:
      type: ApiCall
      parameters:
        method: DELETE
        uri: "{{serviceAccountApiUri}}"
    RemoveModule:
      type: DockerServiceRemove
      parameters:
        name: "{{moduleServiceName}}"
```

CHAPTER 2

Tenants

Tenant in **Servitiom** represents client organization, who using subset of registered services. Tenant simply compose services for unique business purpose. Service instance can be attached to only one tenant and when tenant is removed with all attached resources.

2.1 Adding a Tenant

2.1.1 REST API

Overview

Resource path	/tenants
HTTP Method	POST
Request/Response Format	application/json

Sample request

```
{
    "id": "1",
    "name": "Big Oil Company",
    "customProperties": {
        "clientClass": "A+"
    }
}
```

2.2 Removing a Tenant

2.2.1 REST API

Overview

Resource path	/tenants/{tenantId}
HTTP Method	DELETE
Request/Response Format	application/json

2.3 Upadating a Tenant

2.3.1 REST API

Overview

Sample request

```
{
    "name": "Big Oil Company",
    "customProperties": {
        "clientClass": "A+"
    }
}
```

2.4 Getting details about Tenants

2.4.1 REST API

Overview

Resource path	/tenants
HTTP Method	GET
Request/Response Format	application/json

2.5 Getting details about Tenant

2.5.1 REST API

Overview

Resource path	/tenants/{tenantId}
HTTP Method	GET
Request/Response Format	application/json

CHAPTER 3

Variables