INFN Cloud Use Cases with advanced docker-compose based implementations

Marica Antonacci (INFN BA)

Uso e sviluppo di applicazioni e servizi su INFN Cloud (CLueApp)

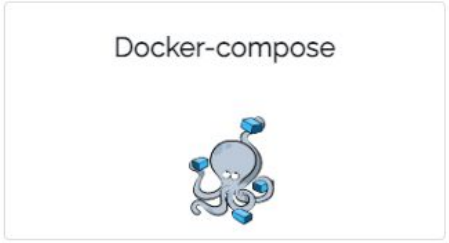
13-16 September 2022

Marica Antonacci (marica.antonacci@ba.infn.it)

INFN Cloud services implementation: TOSCA & Ansible

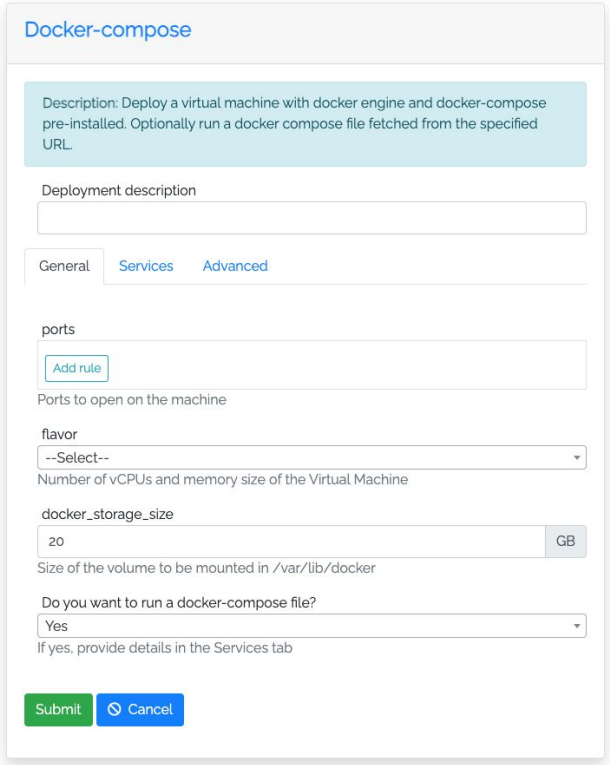
- The INFN Cloud services have been implemented improving and evolving the TOSCA types and templates developed during the INDIGO-DataCloud projects and spin-off (DEEP, EOSC-Hub, etc.)
- **TOSCA** is used to describe the **service topology**
- Each node in a TOSCA template has a specific type
- **Node types in TOSCA have associated implementations that provide the automation** (e.g. in the form of scripts such as Bash, Chef or Python) for the normative lifecycle operations of a node
- **INDIGO-derived types use Ansible recipes**

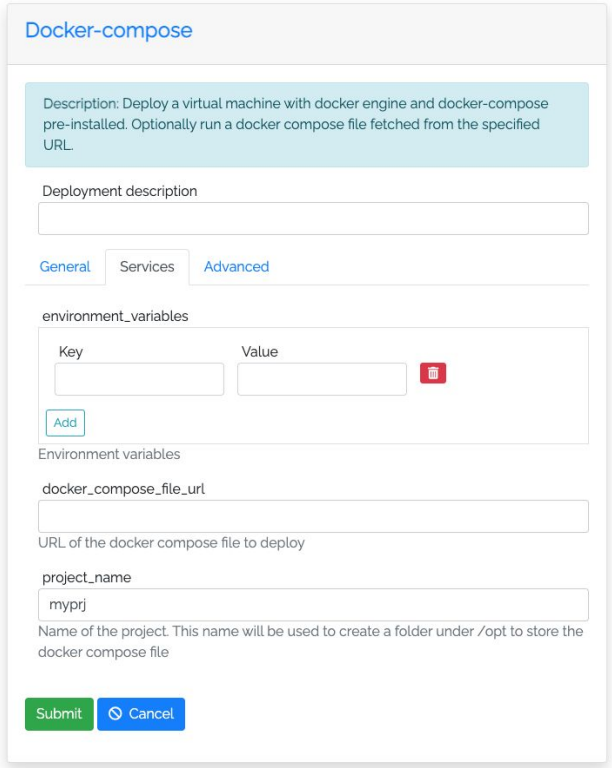
Docker compose base implementation



Let's have a look at the TOSCA template

https://baltig.infn.it/infn-cloud/tosca-templates/-/blob/master/docker/docker_compose.yaml





TOSCA definition

```yaml
docker_compose_service:
  type: tosca.nodes.indigo.DockerCompose
  properties:
    project_name:  { get_input: project_name }
    docker_compose_file_url: { get_input: docker_compose_file_url }
    environment_variables: { get_input: environment_variables }
  requirements:
    - host: server

server:
  type: tosca.nodes.indigo.Compute
  properties:
    os_users: { get_input: users }
  capabilities:
    endpoint:
      properties:
        ports: { get_input: service_ports }
    host:
      properties:
        num_cpus: { get_input: num_cpus }
        mem_size: { get_input: mem_size }
    os:
      properties:
        distribution: ubuntu
        type: linux
        version: 20.04
```

```yaml
tosca.nodes.indigo.DockerCompose:
  derived_from: tosca.nodes.SoftwareComponent
  properties:
    docker_compose_version:
      type: version
      required: no
      default: 1.25.5
    docker_compose_file_url:
      type: string
      required: no
      default: ""
    environment_variables:
      required: no
      default: []
      type: list
      entry_schema:
        type: map
        entry_schema:
          type: string
    project_name:
      type: string
      required: yes
  artifacts:
    docker_role:
      file: indigo-dc.docker,v2.1.3
      type: tosca.artifacts.AnsibleGalaxy.role
  interfaces:
    Standard:
      start:
        implementation: https://baltig.infn.it/infn-cloud/tosca-types/raw/master/artifacts/docker/docker-compose_start.yml
        inputs:
          docker_compose_version: { get_property: [ SELF, docker_compose_version ] }
          docker_compose_file_url:  { get_property: [ SELF, docker_compose_file_url ] }
          project_name:  { get_property: [ SELF, project_name ] }
          environment_variables: { get_property: [ SELF, environment_variables ] }
```

Ansible role

Ansible playbook

https://baltig.infn.it/infn-cloud/tosca-types/-/blob/master/tosca_types/infrastructure/docker_types.yaml

The playbook

```yaml
---
- hosts: localhost
  connection: local
  vars:
    docker_bridge_ip_cidr: "172.0.17.1/24"
  tasks:

    - name: Call Docker role
      include_role:
        name: indigo-dc.docker

    - name: "Create env file, download and start the docker compose file"
      block:

        - name: "create directory path to store the configuration files"
          file:
            path: "/opt/{{ project_name }}"
            state: directory
            mode: 0755

        - name: Set environment variables
          lineinfile:
            path: /opt/{{ project_name }}/.env
            line: "{{ item.key }}={{ item.value }}"
            create: yes
          with_dict: "{{ environment_variables }}"

        - name: Add HOST_PUBLIC_IP and additional environment variables
          lineinfile:
            path: /opt/{{ project_name }}/.env
            line: "{{ item.key }}={{ item.value }}"
            create: yes
          with_items:
            - { key: "HOST_PUBLIC_IP", value: "{% if IM_NODE_PUBLIC_IP is defined %}{{IM_NODE_PUBLIC_IP}}{% else %}{{IM_NODE_PRIVATE_IP}}{%
endif %}" }

        - name: "Download the docker-compose file"
          get_url:
            url: "{{ docker_compose_file_url }}"
            dest: "/opt/{{ project_name }}/docker-compose.yaml"

        - name: "Start the service"
          docker_service:
            project_src: "/opt/{{ project_name }}"
            state: present
      when: docker_compose_file_url != ""
```

1. install docker and compose
2. create the project dir
3. create the .env file with all the envariable variables

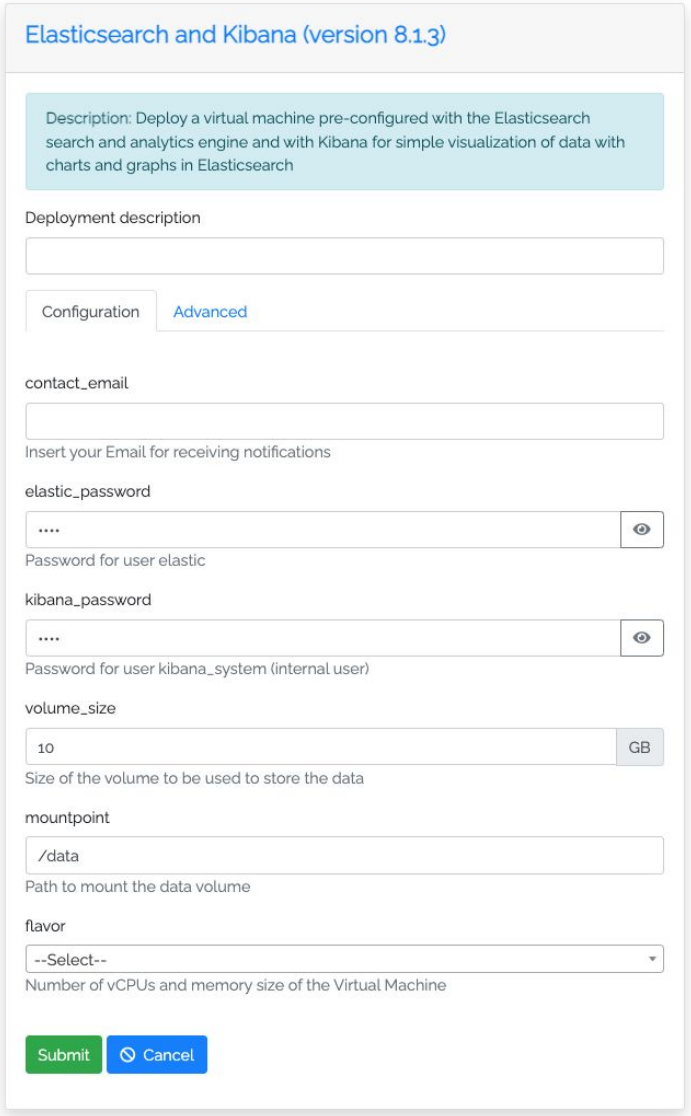
If a docker compose file url is defined:

4. download the docker compose file
5. start the services

EK services implementation

The elasticsearch + kibana (EK) service has been implemented extending the basic docker compose service, deriving the custom type from ***tosca.nodes.indigo.DockerCompose***

EK service implementation



TOSCA template:

https://baltig.infn.it/infn-cloud/tosca-templates/-/blob/master/single-vm/elasticsearch_kibana.yaml

```
docker_compose_service:
  type: tosca.nodes.indigo.DockerCompose.Elastic
  properties:
    project_name: elastic
    environment_variables:
      - ELASTIC_VERSION: "8.1.3"
      - ELASTIC_PASSWORD: { get_input: elastic_password }
      - KIBANA_PASSWORD: { get_input: kibana_password }
      - CERT_EMAIL: { get_input: contact_email }
      - DATA_DIR: { get_input: mountpoint }
  requirements:
    - host: kibana_es_server
```

Derived type

```
tosca.nodes.indigo.DockerCompose.Elastic:
  derived_from: tosca.nodes.indigo.DockerCompose
  properties:
    docker_compose_file_url:
      type: string
      default: https://baltig.infn.it/infn-cloud/tosca-types/raw/master/artifacts/docker/elastic/docker-compose.yml
  artifacts:
    docker_role:
      file: indigo-dc.docker,v2.1.3
      type: tosca.artifacts.AnsibleGalaxy.role
  interfaces:
    Standard:
      configure:
        implementation: https://baltig.infn.it/infn-cloud/tosca-types/raw/master/artifacts/docker/elastic/configure.yml
        inputs:
          project_name:  { get_property: [ SELF, project_name ] }
          environment_variables: { get_property: [ SELF, environment_variables ] }
      start:
        implementation: https://baltig.infn.it/infn-cloud/tosca-types/raw/master/artifacts/docker/docker-compose_start.yml
        inputs:
          docker_compose_version: { get_property: [ SELF, docker_compose_version ] }
          docker_compose_file_url:  { get_property: [ SELF, docker_compose_file_url ] }
          project_name:  { get_property: [ SELF, project_name ] }
```

The property *docker_compose_file_url* is overridden providing the default docker compose file. All other properties are inherited by the parent type

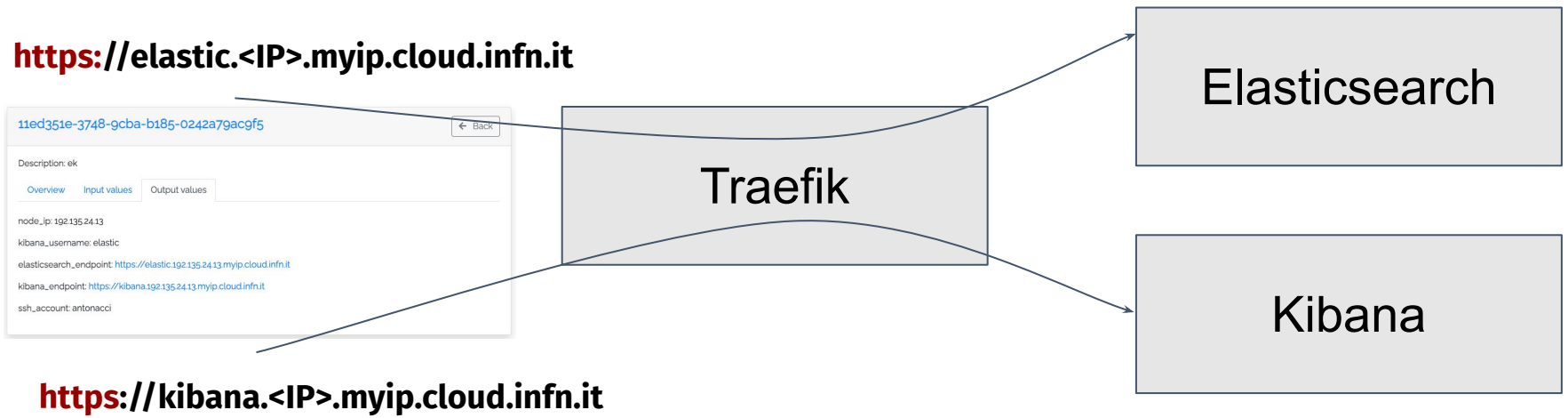
The interfaces are specialised too in order to perform custom preliminary configurations (see next slide)

Customized playbook

```yaml
---
- hosts: localhost
  connection: local
  tasks:
    - name: set timezone to Europe/Rome
      timezone:
        name: Europe/Rome

    - name:
      shell: sysctl -w vm.max_map_count=1048576 && echo "vm.max_map_count = 1048576" > /etc/sysctl.d/30-vm.max_map_count.conf

    - name: "create directory path to store the configuration files"
      file:
        path: "{{ item }}"
        state: directory
        mode: 0755
      loop:
        - "/opt/{{ project_name }}"
        - "/opt/{{ project_name }}/traefik"

    - name: set data dir
      set_fact:
        data_dir: "{{ item.value }}"
      with_dict: "{{ environment_variables }}"
      when: "'DATA_DIR' in item.key"


    - name: "create data directory (if it does not exist)"
      file:
        path: "{{ data_dir }}"
        state: directory
        mode: 0755
        owner: 1000
        recurse: yes

    - name: download tls.toml
      get_url:
        url:  "https://baltig.infn.it/infn-cloud/tosca-types/raw/master/artifacts/docker/elastic/tls.toml"
        dest: "/opt/{{ project_name }}/traefik/tls.toml"
        mode: 0440
```

1. set the time zone
2. adjust kernel settings (see doc)
3. create the needed dirs to host configuration files
4. create the dir to store the collected data
5. download and install the TLS settings for traefik

The docker compose file



Traefik terminates the SSL connections: it is configured to use an ACME provider (Let's Encrypt) for automatic certificate generation.

https://baltig.infn.it/infn-cloud/tosca-types/-/blob/master/artifacts/docker/elastic/docker-compose.yml

Sync&Share service

The INFN-Cloud Sync&Share aaS is currently based on the popular ***ownCloud*** storage solution.
INFN-Cloud users have full control over the configuration parameters of their Cloud Storage instance, as well as on third party accesses to the stored data.
Main features:

- **S3 based Object Storage backend** where data is replicated over two backbone data centers (CNAF, BARI)
- Authentication/Authorization based on **INFN-Cloud IAM** (via OIDC)
- **programmatic access** to user data via Rclone, including remote mount and folder sync
- embedded, automated DB and configuration **backup**
- embedded, pre-configured **monitoring system** with alert notifications

Service implementation

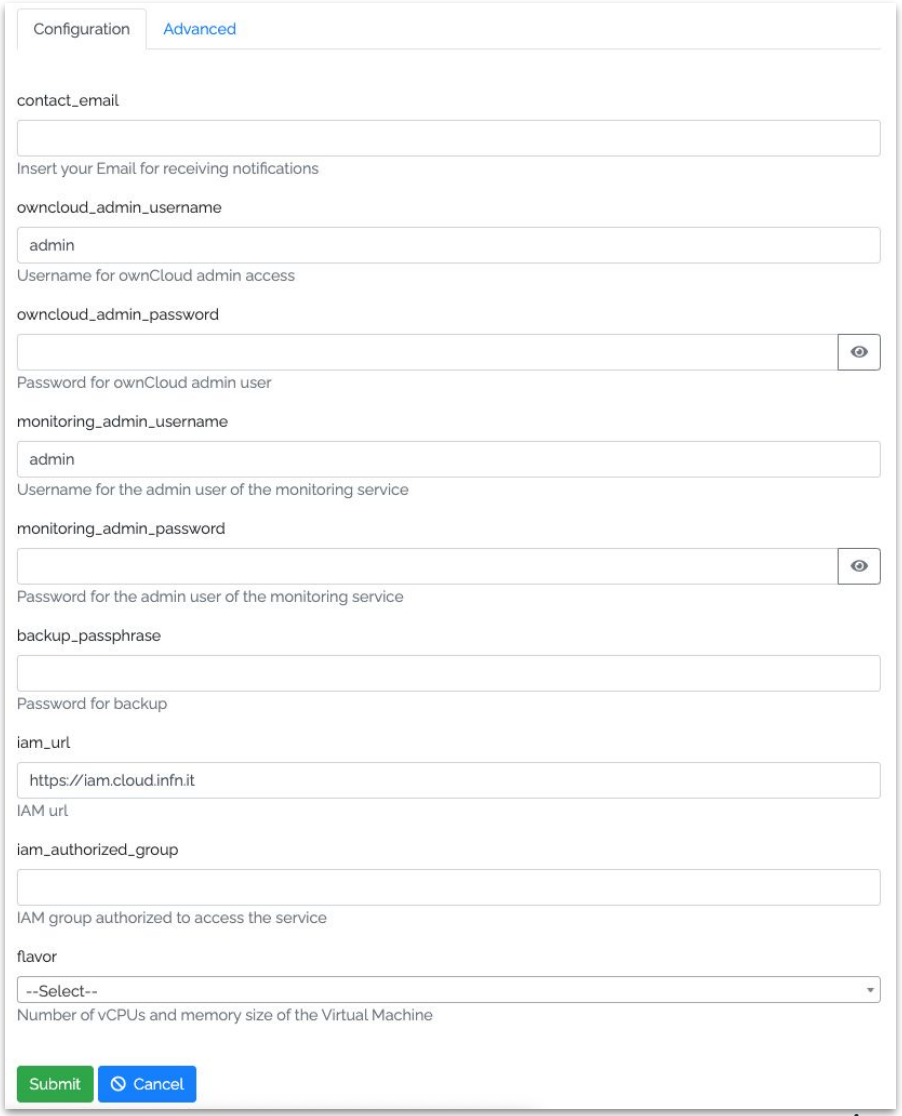
The core setup is based on a docker compose file like the EK service, but in this case the implementation is different.

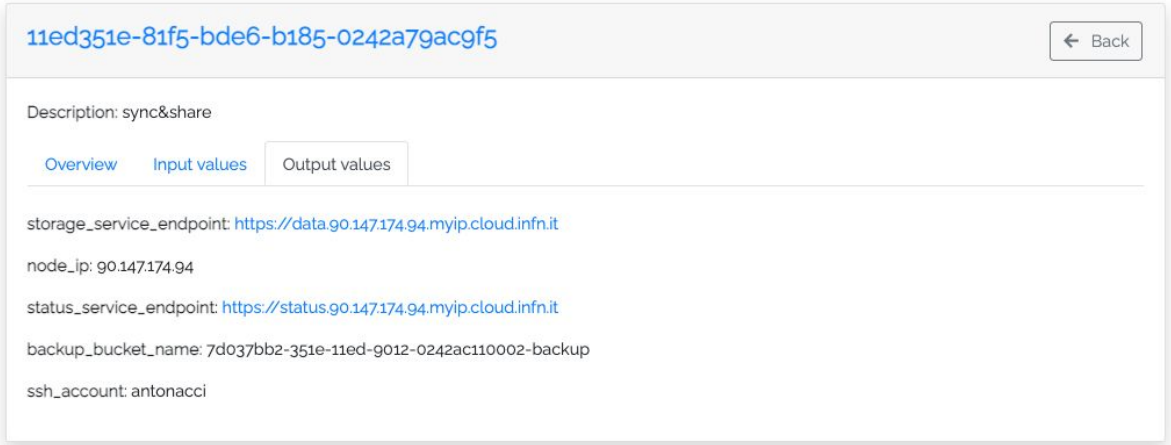
Since the configuration of this service is a bit more complex, we decided not to derive from the tosca.nodes.indigo.DockerCompose.

A new tosca type has been developed as a new Software Component.

Sync&Share aaS

Service configuration and deployment outputs





TOSCA template:

https://baltig.infn.it/infn-cloud/tosca-templates/-/blob/master/single-vm/cloud_storage_service.yaml

TOSCA definition

```yaml
node_templates:

  s3_owncloud_bucket:
    type: tosca.nodes.indigo.S3Bucket
    properties:
      bucket_name: { get_input: owncloud_bucket_name }
      aws_access_key: { get_input: aws_access_key }
      aws_secret_key: { get_input: aws_secret_key }
      s3_url: 'https://s3.cloud.infn.it'
    requirements:
      - host: server

  s3_backup_bucket:
    type: tosca.nodes.indigo.S3Bucket
    properties:
      bucket_name: { get_input: backup_bucket_name }
      aws_access_key: { get_input: aws_access_key }
      aws_secret_key: { get_input: aws_secret_key }
      s3_url: 'https://s3.cloud.infn.it'
    requirements:
      - host: server

  docker_compose_service:
    type: tosca.nodes.indigo.CloudStorageService
    properties:
      owncloud_hostname: { concat: [ "data.",  get_attribute: [ HOST, public_address, 0 ], ".myip.cloud.infn.it" ] }
      nagios_hostname: { concat: [ "status.",  get_attribute: [ HOST, public_address, 0 ], ".myip.cloud.infn.it" ] }
      s3_data_bucket: { get_property: [ s3_owncloud_bucket, bucket_name ] }
      s3_backup_bucket: { get_property: [ s3_backup_bucket, bucket_name ] }
      s3_access_key: { get_property: [ s3_owncloud_bucket, aws_access_key ] }
      s3_secret_key: { get_property: [ s3_owncloud_bucket, aws_secret_key ] }
      s3_endpoint: { get_property: [ s3_owncloud_bucket, s3_url ] }
      owncloud_admin_user: { get_input: owncloud_admin_username }
      owncloud_admin_passw: { get_input: owncloud_admin_password }
      mysql_root_passw: { get_input: mysql_root_password }
      nagios_admin_user: { get_input: monitoring_admin_username  }
      nagios_admin_passw: { get_input: monitoring_admin_password }
      backup_passphrase: { get_input: backup_passphrase  }
      contact_email: { get_input: contact_email }
      smtp_username: { get_input: smtp_username }
      smtp_password: { get_input: smtp_password }
      iam_url: { get_input: iam_url }
      iam_group: { get_input: iam_authorized_group }
    requirements:
      - host: server
      - dependency: s3_owncloud_bucket
      - dependency: s3_backup_bucket
```

S3 storage area for hosting the user data

S3 storage area for hosting the backup data

The new type t***osca.nodes.indigo.CloudStorageService*** is derived from the normative type *tosca.nodes.SoftwareComponent*

The ansible playbook

```yaml
---
- hosts: localhost
  connection: local
  roles:
  - role: ansible-role-cloudstorage
```

https://baltig.infn.it/infn-cloud/tosca-types/raw/master/artifacts/cloudstorage/configure.yml

role repository: https://baltig.infn.it/infn-cloud/ansible-role-cloudstorage

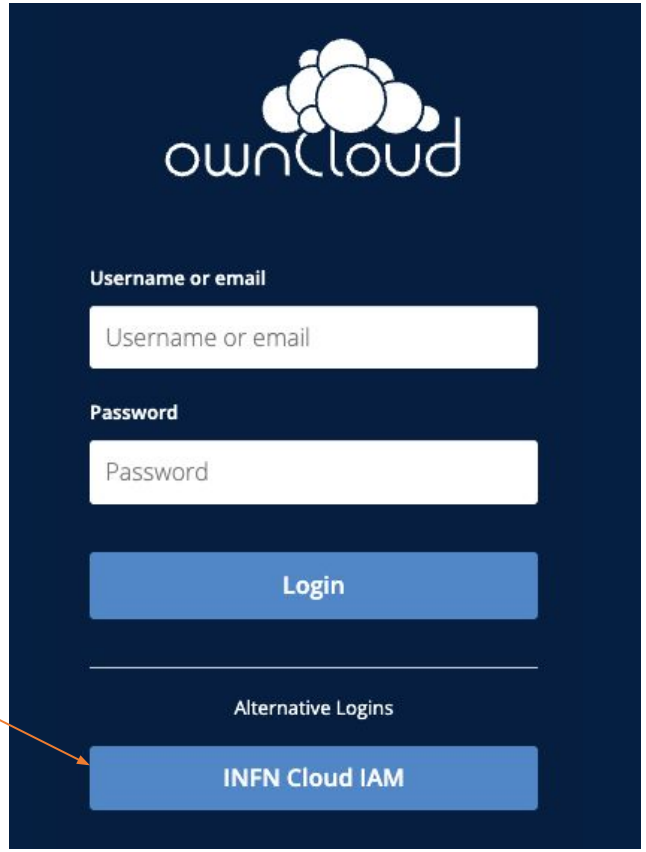
```yaml
- block:
  - name: Register iam client
    uri:
      url: "{{ cloudstorage_iam_register_url}}"
      validate_certs: "no"
      method: POST
      status_code: 201
      headers:
         Content-Type: "application/json"
      body:
        redirect_uris:
          - "https://{{ cloudstorage_owncloud_hostname }}/apps/openidconnect/redirect"
        client_name: "oc-client"
        contacts:
          - "{{ cloudstorage_contact_email }}"
        token_endpoint_auth_method: client_secret_basic
        scope: openid email profile
        grant_types:
          - authorization_code
        response_types:
          - code
      body_format: json
      return_content: yes
    register: iam_response
```

This block of tasks configures the integration with IAM

```yaml
  - name: Save client info
    copy:
      content: "{{iam_response.json}}"
      dest: /opt/{{ cloudstorage_project_name }}/.client-iam.json

  - template:
      src: oidc.config.php.j2
      dest: "/opt/{{ cloudstorage_project_name }}/oidc.config.php"
    vars:
      iam_client_id: "{{iam_response.json.client_id}}"
      iam_client_secret: "{{ iam_response.json.client_secret }}"
  when: not oidc_config.stat.exists|bool
```

```yaml
- name: "Create the docker-compose file"
  copy:
    src: docker-compose.yaml
    dest: "/opt/{{ cloudstorage_project_name }}/docker-compose.yaml"

- name: "Start the service"
  docker_service:
    project_src: "/opt/{{ cloudstorage_project_name }}"
    state: present

- name: "Enable openidconnect app"
  command: docker exec owncloud occ app:enable openidconnect
  register: result
  until: result.rc == 0
  retries: 5
  delay: 60
```

Once the services are up and running, the oidc app is enabled



The docker compose file

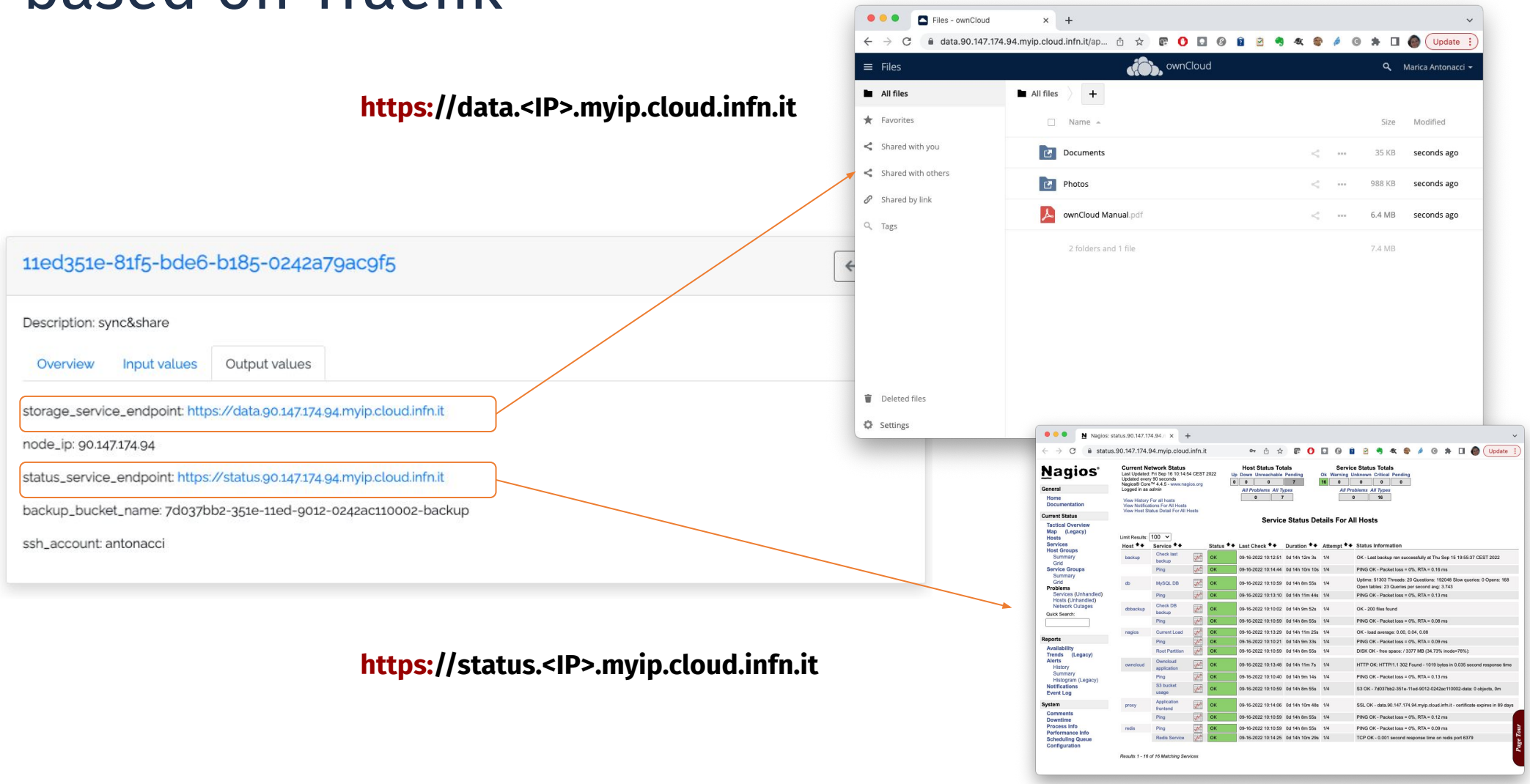
https://baltig.infn.it/infn-cloud/ansible-role-cloudstorage/-/blob/main/files/docker-compose.yaml

```
services:
  proxy:
    container_name: proxy
    image: harbor.cloud.infn.it/cache/library/traefik:${TRAEFIK_VERSION}

  owncloud:
    container_name: owncloud
    image: harbor.cloud.infn.it/cache/owncloud/server:${OWNCLOUD_VERSION}

  redis:
    container_name: redis
    image: harbor.cloud.infn.it/cache/webhippie/redis:latest


  db:
    container_name: db
    image: harbor.cloud.infn.it/cache/library/mariadb:10.5.11


  nagios:
    container_name: nagios
    image: harbor.cloud.infn.it/library/storageservice-nagios
```

```
  backup:
    image: harbor.cloud.infn.it/library/storageservice-backup
    container_name: backup
    volumes:
      - files:/backup/files
      - backup:/backup/db
      - backup-logs:/backup/logs
      - nagios-conf:/backup/nagios/conf
      - nagios-data:/backup/nagios/data
      - letsencrypt:/backup/letsencrypt


  dbbackup:
    image: harbor.cloud.infn.it/library/storageservice-mariadb-backup
    container_name: dbbackup
    volumes:
      - mariadb:/var/lib/mysql
      - backup:/backup
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro
    environment:
      - CRON_SCHEDULE=10 15 * * *
```

Services are accessed through the reverse proxy based on Traefik

https://data.<IP>.myip.cloud.infn.it



https://status.<IP>.myip.cloud.infn.it

References:

User guides:

- Docker compose: https://guides.cloud.infn.it/docs/users-guides/en/latest/users_guides/howto7.html
- Elasticsearch+Kibana: https://guides.cloud.infn.it/docs/users-guides/en/latest/users_guides/howto5.html
- Sync&Share aaS: https://guides.cloud.infn.it/docs/users-guides/en/latest/users_guides/howto11.html

Thank you

for your attention!