CrossplaneProvider for Taikun Workshop

Contents

1 Introduction

The purpose of this workshop is to introduce you to Crossplane and the Crossplane Provider for Taikun. The latter will allow you to use Crossplane to manage resources in Taikun.

2 How to read this document

- Text in this form is to be typed, as is, on the command line.

```
cd workshop/
ls
echo Hello!
```

- This form of text shows screen output, usually the output of commands.

```
task_00/
task_01/
...
Hello!
```

- This format is for code in Crossplane's configuration filess,

```yaml
apiVersion: organization.taikun.jet.crossplane.io/v1alpha1
kind: Organization
metadata:
  name: orga-raph-crossplane
spec:
  forProvider:
    name: "new-orga"
    fullName: "Create taikun organization with crossplane"
    discountRate: 42
    city: "Praha"
    billingEmail: "billing2@foo.org"
    email: "contact@foo.org"
    phone: "065100035103"
  providerConfigRef:
    name: default
```

3 Setup

To complete this workshop, you will need to install the Taikun Crossplane provider and the workshop. You need also a working kubernetes cluster with its config file.

3.1 Requirements

- You must have Go version 1.17 or newer installed.
- You will need Git to clone the provider's repo.
- You will need Taikun and Openstack credentials.

3.2 Creating a cluster

To create a kubernetes cluster you can follow the steps of the following link. https://docs.taikun.cloud/guidelines/creating-a-cluster/

3.3 Installing Go

This Goland tutorial explains how to install Go on OS X, Windows and Linux.

3.4 Downloading the provider repository

In a first console window clone the provider directory.

```
git clone https://github.com/itera-io/provider-jet-taikun.git
cd provider-jet-taikun/
```

3.5 Downloading the workshop files

Clone the workshop directory in another console window and switch into the `workshop/` directory.

```
git clone https://github.com/itera-io/provider-jet-taikun-workshop.git
cd provider-jet-taikun-workshop/workshop/
```

4 Documentation

The provider documentation is available in ‘docs‘ directory of the provider repository.

5 Tasks

The end goal of this workshop is to have an operational Taikun project built solely with Crossplane configuration files. By following a step by step process, you will discover how various Taikun resources are declared and managed using Crossplane.

All your work will be done in the `workshop/` directory. These are its initial contents.

```
./workshop/
|-- providerconfig.yaml
|-- taikun_secret.yaml.tmpl
```

`providerconfig.yaml` contains the Provider configuration, namely its source address and what credentials to use. You will not need to edit this file.

```yaml
apiVersion: taikun.jet.crossplane.io/v1alpha1
kind: ProviderConfig
metadata:
    name: providerconfig-workshop
spec:
    credentials:
        source: Secret
        secretRef:
            name: my-creds
            namespace: crossplane-system
            key: credentials
```

During this workshop, each task should be coded in a separate config file. At the end of the workshop, your directory will be organized as such:

```
./workshop/
|-- providerconfig.yaml
|-- taikun_secret.yaml.tmpl
|-- taikun_secret.yaml
|-- openstack_secret.yaml.tmpl
|-- openstack_secret.yaml
|-- organization.yaml
|-- kubeprofile.yaml
|-- slack_alerting.yaml
|-- cloudcred.yaml
|-- user_ap.yaml
|-- project.yaml
```

5.1 Provider and cluster setup

5.1.1 Authentification

In order to complete the tasks that follow, you will need to provide Taikun credentials to Crossplane. You will need a Partner account as some of the tasks, such as creating an organization, require Partner privileges.

Input variables will be explained later in the workshop. For now, simply edit `taikun_secret.yaml.tmpl` and replace the values of `email` and `password` with your Taikun credentials.

```
# taikun_secret.yaml.tmpl
"email": "your_email",
"password": "your_password",
```

Then you can run the following command to create a secret which stores your Taikun credentials to which the providerconfig refers.

```
TAIKUN_EMAIL="your-email"
TAIKUN_PASSWORD="your-password"
cat taikun_secret.yaml.tmpl | sed -e "s/your_email/${TAIKUN_EMAIL}/g" \
| sed -e "s/your_password/${TAIKUN_PASSWORD}/g" > taikun_secret.yaml
```

To find out more about providing sensitive data in Kubernetes, see this page.

5.1.2 Prepare the cluster to use the provider

First in the console window where you have clone the provider repository, please write the following command.

```
make submodules
```

Then this one to create the crd's:

```
kubectl apply -f package/crds
```

Now you can write in the same console window these commands to run against your kubernetes cluster.

```
make run
```

If all is working well the previous command is blocking. Open a new window and execute the following command to create a new namespace named `crossplane-system` in your cluster.

```
kubectl create namespace crossplane-system --dry-run=client -o yaml | kubectl apply -f -
```

In order to connect to taikun with the Crossplane provider you can now execute the following command:

```
kubectl apply -f examples/providerconfig/
```

Important

Be sure that you have your kubernetes config file of your cluster in your `/.kube/` directory. Otherwise please create this directory and paste your config file like this : `/.kube/config`.
This place is normally the default one where the kubectl command will look for a kubernetes cluster config. Otherwise you will have may be to set the KUBECONFIG environment variable to this location:

```
export KUBECONFIG="~/.kube/config"
```

5.2 Task 0: Organization

> **Note**
>
> For this task, please write your code in the file `organization.yaml` at the root of the `workshop/` directory.

This objective of this first task is to create an organization. All resources created in the future will be part of this organization. As this is the first task, every step of the process is documented.

Before you do anything, start by preparing your working directory for other commands. Be sure that the section 3 *Setup* is done before and the command `make run` is blocking and run well.
If all went well, you should see something like the following message.

```
15:41:54 [ .. ] go build linux_amd64
go: downloading ...
go: downloading ...
[...]
15:42:40 [ OK ] go build linux_amd64
15:42:40 [ .. ] Running Crossplane locally out-of-cluster . . .
/home/nivaultr/Documents/Itera/workshop/test-cross-workshop/provider-jet-taikun/_output/
bin/linux_amd64/provider --debug
1.6674001605709026e+09 DEBUG provider-jet-taikun Starting {"sync-period": "1h0m0s"}
I1102 15:42:41.683337  466564 request.go:665] Waited for 1.036556627s due to client-side
throttling, not priority and fairness, request:
GET:https://185.22.97.151:6443/apis/project.taikun.jet.crossplane.io/v1alpha1?timeout=32s
1.6674001619435318e+09 INFO controller-runtime.metrics Metrics server is starting to
listen {"addr": ":8080"}
1.667400161956083e+09 INFO Starting server {"path": "/metrics", "kind": "metrics",
"addr": "[::]:8080"}
1.6674001619562776e+09 INFO controller.managed/accessprofile.taikun.jet.crossplane.io/
v1alpha1, kind=profile  Starting EventSource {"reconciler group": "accessprofile.
taikun.jet.crossplane.io", "reconciler kind": "Profile", "source": "kind source:
*v1alpha1.Profile"}
1.6674001619563134e+09 INFO controller.managed/accessprofile.taikun.jet.crossplane.io/
v1alpha1, kind=profile  Starting Controller {"reconciler group": "accessprofile.
taikun.jet.crossplane.io", "reconciler kind": "Profile"}
[...]
```

Once all this steps has been done correctly, you can declare your organization resource. Create `organization.yaml` and write the following configuration block to it.

```yaml
apiVersion: organization.taikun.jet.crossplane.io/v1alpha1
kind: Organization
metadata:
    name: myorg
spec:
    forProvider:
        name: <name>
        fullName: <full-name>
        discountRate: 42
    providerConfigRef:
        name: providerconfig-workshop
```

Be sure to replace `<name>` and `<full-name>` with names of your choosing. You can also choose another metadata name instead of `myorg`.

Tip

Notice the syntax of the configuration block, as you are creating a resource, it begins with the keyword `apiVersion`, followed by its CRD name and the api version. The type of resource is always lowercase and followed by the name of the provider, thus `"organization.taikun.jet.crossplane.io/v1alpha1"`. Following the resource's `apiVersion` is a metadata name, it must be unique for this type of resource, and is used to refer to this specific resource, as you will find out later. Watch out, this field does not correspond to the name of the resource in Taikun.

Three arguments are then defined in `spec` then `forProvider` fields: `name`, `fullName` and `discountRate`. On the left side of the colon is the argument's identifier, on the right is its value. See the documentation of Taikun's organization resource in the docs directory of the provider repository for a full list of arguments, i.e. the resource's *schema*.

Metadata names and argument names can contain letters, digits, underscores and hyphens and may not start with a digit. Their length must be lower than 30 character.

Now apply your changes.

Tip

If you have already created resources, `kubectl apply` will create a new resource by making request to Taikun's API.

```
kubectl apply -f organization.yaml
```

To check if your resource is created successfully execute the following command :

```
watch kubectl get managed
```

You should get a False state.

```
NAME                                                    READY   SYNCED   EXTERNAL-NAME   AGE
organization.organization.taikun.jet.crossplane.io/myorg   False   False                    15s
```

In order to show the errors describe the resource you have just created.

```
kubectl describe organizations.organization.taikun.jet.crossplane.io myorg
```

You should have in the `Event` field at the bottom of the output an error message like the following:

```
TO DO
```

Tip

`kubectl describe` command is used to describe (obviously) the resource you created. You can use it by different way:

```
kubetcl describe crd-name my-ressource
kubectl describe -f path-to-my-ressource-file
```

To get the name of the crd you can use the `kubectl get crds` command or the `kubectl get crds | grep taikun` to see all taikun crds.

Now fix the discount rate so it is in the range 0-100 and run `kubectl apply` once more. Normally you should have the following output for `watch kubectl get managed` command.

```
NAME                                                          READY   SYNCED   EXTERNAL-NAME   AGE
organization.organization.taikun.jet.crossplane.io/myorg      True    True     14236           15s
```

Tip

The `EXTERNAL-NAME` field is the id of your resource in Taikun.

You may wish to check the organization was indeed created at app.taikun.cloud/organizations.

5.3 Task 1: Kubernetes Profile

Note

For this task, please write your code in the file `kubeprofile.yaml` at the root of the `workshop/` directory.

Now that you have created an organization, you will create a Kubernetes profile belonging to it. Check the kubernetes profile resource's schema on the provider's documentation in the `docs/` directory in the provider repository. and declare the resource in `kubernetesprofile.yaml`. Set `organizationId` to the ID of the organization created in the previous task (see subsection 5.2 *Task 0: Organization*).

Tip

To find the ID of the organization, run

```
watch kubectl get managed
```

Then you should get the `EXTERNAL-NAME` of the `myorg` organization.

```
NAME                                                          READY   SYNCED   EXTERNAL-NAME   AGE
organization.organization.taikun.jet.crossplane.io/myorg      True    True     14236           15s
```

So here the organization ID will be `14236`.

Feel free to set some of kubernetes profile's other optional attributes, such as `scheduleOnMaster` and `loadBalancingSolution`.

Once you have declared your resource, apply and move on to the next task.

5.4 Task 2: Slack Configuration & Alerting Profile

Note

For this task, please write your code in the file `slack_alerting.yaml` at the root of the `workshop/` directory.

You will now create an alerting profile using a Slack configuration.

1. Start by declaring a Slack configuration. You can found its documentation in the `docs/` directory in the provider repository.

   Its hook URL should be `https://hooks.myapp.example/ci`. It must send **alert-type notifications only** to the channel `ci`.

2. You can now declare the alerting profile. Here is its documentation.
   The alerting profile should send notifications **daily** using the Slack configuration declared above.

Important

As always, your resources should belong to the organization created in subsection 5.2 *Task 0: Organization*.

But this time we will use the Reference arguments. Indeed as it is said in the documentation of alerting profile resource you have a `organizationIdRef` arguments and the latter allows to refer to an organization by his metadata name.

For instance, the previous metadata name of the organization you have created is `myorg` (or the name you have chosen). So the `organizationIdRef` argument value will be `myorg` (or the name you have chosen).

Now you can do the same thing for the argument which refer to the alerting profile resource.

Tip

You can write configuration for multiple resource in one file, you just have to separate the resources configurations with `"--"`.

```
apiVersion: organization.taikun.jet.crossplane.io/v1alpha1
kind: Organization
metadata:
    name: myorg1
spec:
    [...]
---
apiVersion: organization.taikun.jet.crossplane.io/v1alpha1
kind: Organization
metadata:
    name: myorg2
spec:
    [...]
```

Once you have declared these two new resources, apply and move on to the next task.

5.5 Task 3: Cloud Credentials

Note

For this task, please write your code in the file `cloudcred.yaml` at the root of the `workshop/` directory.

Important

You will need OpenStack credentials to complete this task.

Cloud credentials are needed to create a Taikun project. In a real work environment, cloud credentials should not be stored under version control; it's why we'll use kubernetes secret.

Define the OpenStack cloud credential resource in `cloudcred.yaml`. Here is its documentation.
You will need a secret to provide sensitive data like your Openstack password. For this, create a new file named `openstack_secret.yaml` to create a secret configuration for Openstack cloud credentials.

Here is a secret template :

```
apiVersion: v1
kind: Secret
metadata:
    name: <secret-name>
    namespace: <namespace>
type: Opaque
data:
    password: <openstack-password-base64>
```

- The `<secret-name>` argument is the name you will refer to during the creation of your Opensatck cloud credential configuration file.
- The `<namespace>` argument is the namespace where your secret will be stored.
- The `<openstack-password>` argument is your Openstack password in base 64.

Tip

To encode your password to base64 you can execute this command

```
    echo -n "your-password" | base64
```

or you can go in a base64 translator like this one

Once you have declared your new resource, apply and move on to the next task.

Important

As always, your resources should belong to the organization created in subsection 5.2 *Task 0: Organization*.

5.6 Task 4: Users

> **Note**
>
> For this task, please write your code in the file `user_ap.yaml` at the root of the `workshop/` directory.

You will now add an user and an alerting profile to the Taikun organization.

You can now declare the resources in `user_ap.yaml`, the resources must belong to the organization created in subsection 5.2 *Task 0: Organization*. Here is user resource documentation and here for the alerting profile

> **Note**
>
> There is multiple kind of role for a Taikun user. Choose the adapted role for your user. For more information, see this page.

Once you have declared the user and access profile resources, apply and move on to the next task.

5.7 Task 5: Project and User attachment

Note

For this task, please write all your code in the file `project.yaml` at the root of the `workshop/` directory.

5.7.1 Project

Finally, you can declare a project resource. In order to create the resource you will need to import some flavor and images to create a kubernetes cluster or a vm. In our case we will create a kubernetes cluster so we just need flavors.

Tip

Please read the this page. It explains how to create a cluster with Taikun and what are the needed resources.

Now it is your turn to create the project with the resources we created in the previous tasks. You can use reference arguments because all the resources needed have been created in your `crossplane-system` namespace in your cluster.

You can found here the project resource documentation.

5.7.2 Project User Attachment

Now, as you just write the configuration file your project, you can assign users to your project. The users you want to attach to the project must be in the same organization. In our case the resources must belong to the organization created in subsection 5.2 *Task 0: Organization*.

So see to the Project User Attachment documentation here and after writing the two resources in `project.yaml` you can apply your file.

6 Resume

Congratulations !! You just finished the workshop !

You can now try to create other resources to be more familiar with the Taikun Crossplane provider.

There is some interesting links that can be useful:

- The provider documentation: https://github.com/itera-io/provider-jet-taikun/tree/main/docs.
- Taikun documentation: https://itera.gitbook.io/taikun/.

Note

To delete a resource you just have to run

```
kubectl delete -f path-to-resource.yaml
```