

OCI – Functions: Quick Start

Setup your OCI environment to start using OCI
Functions

February 2025, Version 3.0
Florian Bonneville

1 Table of Contents

1	Table of Contents	2
2	Create a Virtual Cloud Network	3
3	Create the Function_Server instance	3
3.1	Launch instance	3
3.2	Configure instance	3
4	Create Dynamic Groups	4
4.1	Dynamic Group for the Function Server	4
4.2	Create a Dynamic Group for the Function Server.....	4
4.3	Create a Dynamic Group for the function	4
5	Create OCI Policy	5
6	Create an Auth Token	6
6.1	Go to your account profile	6
6.2	Generate a new Auth Token	6
6.3	Save your Auth Token	6
7	Create and deploy your function application	7
7.1	Create application from the OCI Console	7
7.2	Enable Function logging	7
7.3	Configure your Function_Server using "Local setup".....	7
7.4	Shell Output:	9
7.5	Output from your OCI console :	9
8	Update Dynamic Group and Policy	10
8.1	Retrieve Function OCID	10
8.2	Update DG_FUNCTIONS	10
9	Setup Event Service	11
9.1	Create a rule to detect 'Instance - Launch End' 11	
10	Review Function logs:	12

2 Create a Virtual Cloud Network

In this scenario I create a dedicated VCN because my function doesn't need to communicate with my other applications.

This VCN will host two components:

- An OCI compute instance (VM) which will be used to publish my function
- The OCI Function

Start the VCN Wizard:

- Networking > Virtual Cloud Network

3 Create the Function_Server instance

3.1 Launch instance

Start compute instance creation

- Compute > Instances
- Use an Oracle Linux image
- You can select any shape available
- Connect the instance to the public subnet of the VCN created previously

3.2 Configure instance

Connect to the instance using SSH.

These steps will configure an Oracle Linux instance with all the components required to build your Dev environment.

Connect to your instance using ssh:

```
sudo dnf update -y
sudo dnf install git -y
git clone https://github.com/Olygo/OCI-FN_IMDS-Watcher.git
python3 -m pip install pip --upgrade --user
python3 -m pip install wheel --upgrade --user
python3 -m pip install oci --upgrade --user
python3 -m pip install oci-cli --upgrade --user
sudo dnf config-manager --add-repo=https://download.docker.com/linux/centos/docker-ce.repo -y
sudo dnf install -y docker-ce --nobest -y
sudo systemctl enable docker.service
sudo systemctl start docker.service
sudo usermod -a -G docker opc
curl -LSs https://raw.githubusercontent.com/fnproject/cli/master/install | sh
fn version
shutdown -r now
```

4 Create Dynamic Groups

Dynamic groups will be used to authenticate the FN_Server and the function across OCI APIs.

Create 2 Dynamic Groups:

- **DG_FN_SERVER**
 - o This dynamic group will authenticate the Function Server
- **DG_FUNCTIONS**
 - o This dynamic group will authenticate your OCI Function

4.1 Dynamic Group for the Function Server

Create a dynamic group (e.g. DG_FN_SERVER) that includes the compute instance hosting your Function Server.

If you don't create a dynamic group and the appropriate policy, you must then create an API-key attached to your user account and configure your Function Server using

"oci setup config" command.

4.2 Create a Dynamic Group for the Function Server

- Identity > Domains > your identity domain > Dynamic groups
- Add the following rule **using the OCID of the compute instance hosting your function server**

```
ALL {instance.id = 'ocid1.instance.oc1.xxxxxxxxxxxxxxxxxx'}
```

4.3 Create a Dynamic Group for the function

Now, create a dynamic group (e.g. DG_FUNCTIONS) to allow the function authenticating to OCI APIs

If you don't create a dynamic group and the appropriate policy, your function will not have the proper rights to manage your OCI resources.

- Add the following rule **using the OCID of your function**

```
ALL {resource.id = 'ocid1.fnfunc.oc1.xxxxxxxxxxxxxxxxxx'}
```

<https://docs.oracle.com/en-us/iaas/Content/Functions/Tasks/functionsaccessingociresources.htm>

Because the function has not been created yet, paste above example, then you will update the Dynamic group with the real function' ocid after its creation.

You will get the function ocid through:

- Function > Applications > ***function_app_name*** > ***function_name***

If you have multiples Functions in the same compartment, you can use the following statement:

- Identity & Security > Identity > Dynamic Groups
- Add the following rule **using the OCID of your Function compartment**

```
All {resource.type = 'fnfunction', resource.compartment.id = 'ocid1.compartment.oc1.xxxxxxx'}
```

<https://docs.oracle.com/en-us/iaas/Content/Identity/dynamicgroups/Writing Matching Rules to Define Dynamic Groups.htm>

5 Create OCI Policy

- Identity & Security > Identity > Policies
- **IN THE ROOT COMPARTMENT:**
 - To allow the DG_FN_SERVER dynamic group access to function resources, network resources, and Oracle Cloud Infrastructure Registry (OCIR) at the tenancy level

```
Allow service FaaS to read repos in tenancy
Allow service FaaS to use virtual-network-family in tenancy

Allow dynamic-group 'YourIdentityDomain'/'DG_FN_SERVER' to manage functions-family in tenancy
Allow dynamic-group 'YourIdentityDomain'/'DG_FN_SERVER' to manage repos in tenancy
Allow dynamic-group 'YourIdentityDomain'/'DG_FN_SERVER' to use virtual-network-family in tenancy

Allow dynamic-group 'YourIdentityDomain'/'DG_FUNCTIONS' to manage instance-family in tenancy
```

- **IN A CHILD COMPARTMENT:**
 - To restrict access to a specific compartment

```
Allow service FaaS to read repos in compartment xxxx
Allow service FaaS to use virtual-network-family in compartment xxxx

Allow dynamic-group 'YourIdentityDomain'/'DG_FN_SERVER' to manage functions-family in compartment xxxx
Allow dynamic-group 'YourIdentityDomain'/'DG_FN_SERVER' to manage repos in compartment xxxx
Allow dynamic-group 'YourIdentityDomain'/'DG_FN_SERVER' to use virtual-network-family in compartment xxxx

Allow dynamic-group 'YourIdentityDomain'/'DG_FUNCTIONS' to manage instance-family in compartment xxxx
```

Adapt the scope of your statements according to your security constraints.

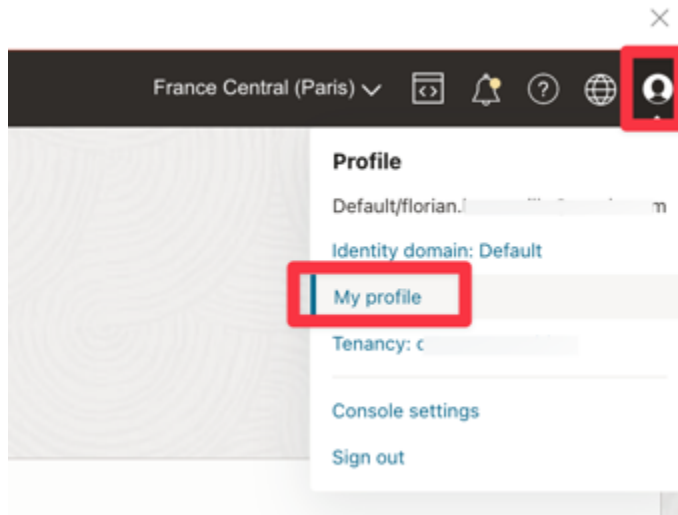
<https://docs.oracle.com/en-us/iaas/Content/Functions/Tasks/functionsrestrictinguseraccess.htm>

6 Create an Auth Token

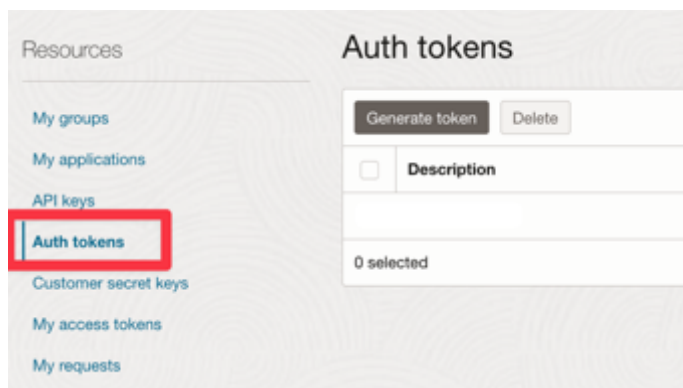
The Auth Token will be used to connect to your Container Registry (OCIR) which will host your docker container.
This token will be required later for the docker login command.

6.1 Go to your account profile

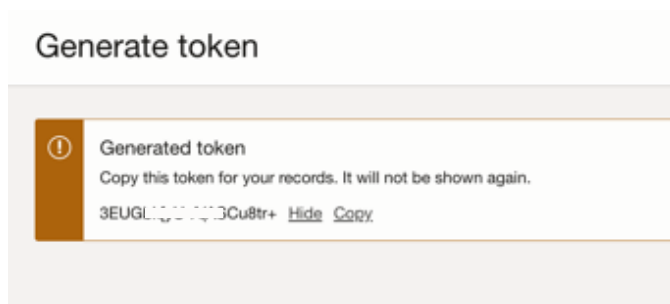
- Click Profile (top right) > My profile



6.2 Generate a new Auth Token



6.3 Save your Auth Token



Save your auth token in your notes or personal vault for later use.

7 Create and deploy your function application

7.1 Create an application from the OCI Console

- Developer Services > Functions > Applications

Create application

Name
My_Functions

VCN in Sys_Functions [\(Change compartment\)](#)
VCN_Functions

subnets in Sys_Functions [\(Change compartment\)](#)
public subnet-VCN_Functions (Regional) ✕

7.2 Enable Function logging

- Developer Services > Functions > Applications > YourApplication

Resources

- Getting started
- Functions
- Configuration
- Signature verification
- Metrics
- Logs**
- Traces

Logs

Category	Status	Log Name	Log Group	Enable Log
Function Invocation Logs	Active	FBO_Functions_invoke	Default_Group	<input checked="" type="checkbox"/> Enabled

7.3 Configure your Function_Server using “Local setup”

My_Functions

Move application Add tags Delete

Application information Tags

General information

OCID: ...g4cag2h4 [Show](#) [Copy](#)
Compartment: Sys_Functions
Logging policy: None
Trace name: None
Created: Wed, Mar 22, 2023, 23:34:56 UTC
Last updated: Wed, Mar 22, 2023, 23:34:56 UTC
Signature verification: Disabled

Network information

Subnets: [public subnet-VCN_Functions](#)
Network security groups: None [Add](#)

Getting started

Cloud Shell setup
Quickly create, deploy and invoke functions using Cloud Shell

Local setup
Set up a development machine to create, deploy and invoke functions ✓

Steps 1 & 2 are not required here **if you have previously downloaded** the function's code through git clone above.

Jump into the function folder

- `cd ./OCI-FN_IMDS-Watcher`

Now run function setup commands from the **Local Setup** section:

Step 3: Context name can be anything:

/!\ DO NOT USE: `--provider oracle` USE: `--provider oracle-ip`

- **`--provider oracle`:**
 - authenticates a user with a local `oci-cli` config file
- **`--provider oracle-ip`:**
 - authenticates an instance with `instance_principals` (through a `dynamic-group`)

```
fn create context YourContextName --provider oracle-ip
fn use context YourContextName
```

Step 4: Compartment hosting your function (use tenancy `ocid` for Root compartment), API endpoint reflects your OCI region:

```
fn update context oracle.compartment-id ocid1.tenancy.oc1..xxxxxxxxxxx
fn update context api-url https://functions.RegionName.oraclecloud.com
```

Step 5: Choose a name for the container registry repository (OCIR), it can be anything **in lower cases without spaces**: `[repo-name-prefix]`

```
fn update context registry RegionCode.ocir.io/TenantNamespace/my-repo-name
```

Step 6: log into OCIR using `docker login` and your **Auth Token** created previously:

```
docker login -u 'TenantNamespace/YourIdentityDomain/me@company.com' RegionCode.ocir.io
```

Step 7: Push the local function code to your OCI Application and OCIR repository:

```
Fn deploy --app YourApplicationName
```


7.4 Shell Output:

```
[opc@inst-func-build-142296 OCI-FN_ ] $ fn create context Sys_Functions --provider oracle-ip
Successfully created context: Sys_Functions
[opc@inst-func-build-142296 OCI-FN_ ] $ fn use context Sys_Functions
Now using context: Sys_Functions
[opc@inst-func-build-142296 OCI-FN_ ] $ fn update context oracle.compartment-id ocid1.compartment.oc1..aaaaaaaabn
Current context updated oracle.compartment-id with ocid1.compartment.oc1..aaaaaaaabnynka5s
[opc@inst-func-build-142296 OCI-FN_ ] $ fn update context api-url https://functions.eu-paris-1.oraclecloud.com
Current context updated api-url with https://functions.eu-paris-1.oraclecloud.com
[opc@inst-func-build-142296 OCI-FN_ ] $ fn update context registry cdg.ocir.io/a/ demo-repo
Current context updated registry with cdg.ocir.io/a/ demo-repo
[opc@inst-func-build-142296 OCI-FN_TagCompute_DT]$ docker login -u ' /florian .com' cdg.ocir.io
Password:
WARNING! Your password will be stored unencrypted in /home/opc/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[opc@inst-func-build-142296 OCI-FN_ ] $ fn deploy --app My_Functions
Deploying oci-fn_ to app: My_Functions
Bumped to version 0.0.1
Using Container engine docker
Building image cdg.ocir.io/ /demo-repo/oci-fn_ 0.0.1 .....
Parts: [cdg.ocir.io demo-repo oci-fn_ 0.0.1]
Using Container engine docker to push
Pushing cdg.ocir.io/ /demo-repo/oci-fn_ :0.0.1 to docker registry...The push refers to repository [cd
753aa253c2e1: Pushed
1466c8241698: Pushed
eb5011488c70: Pushed
64297c36ab1d: Pushed
0059976a0ea6: Pushed
e76ab7648bea: Pushed
f46abc7ec396: Pushed
0.0.1: digest: sha256:4bbf2fff28d8da9df79e70 :3ceaba size: 1782
Updating function oci-fn_ using image cdg.ocir.io/ /demo-repo/oci-fn_ 0.0.1...
Successfully created function: oci-fn_ with cdg.ocir.io/ /demo-repo/oci-fn_ 0.0.1
```

7.5 Output from your OCI console :

Your function has been published into your Application:

The screenshot shows the OCI Functions console interface. On the left is a sidebar with navigation links: 'Getting started', 'Functions' (selected), 'Configuration', 'Signature verification', 'Metrics', and 'Logs'. The main content area is titled 'My_Functions' and includes a green circular icon with a white 'A' and the status 'ACTIVE'. Below the icon are buttons for 'Move application', 'Add tags', and 'Delete'. The 'Application information' tab is active, displaying details such as OCID, Compartment, Logging policy, Trace name, Created, Last updated, and Signature verification status. To the right, the 'Network information' section shows subnets and network security groups. At the bottom, the 'Functions' section contains a table with columns: Name, Image, Image digest, and Invoke endpoint. The table lists a function named 'oci-fn_tagcompute_dt' with its corresponding image, digest, and invoke endpoint.

Name	Image	Image digest	Invoke endpoint
oci-fn_tagcompute_dt	cdg.ocir.io/ /demo-repo/oci-fn_	sha256: 79bc4ffcbba	https://v1.functions.eu-paris-1.oraclecloud.com/20181201/functions/ocid1.fnfunc.oc1.eu-paris-1.aaaaaaaame/cxvmtxvzyzbuyq/actions/invoke

Your container function has been stored into your container registry:

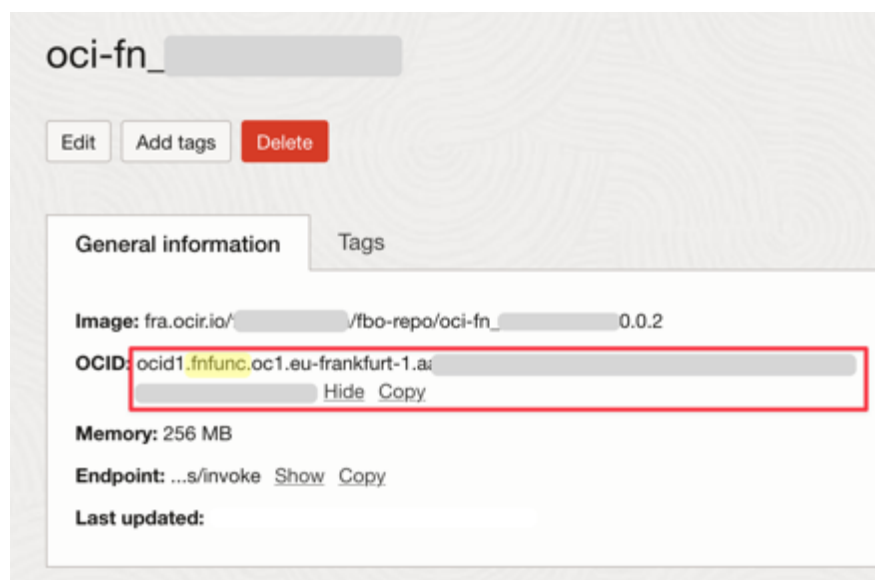
- Containers & Artifacts > Containers & Artifacts > Container Registry



By default, the repository is created in the root compartment, you can move it to another compartment using: Action > Move compartment.

8 Update Dynamic Group and Policy

8.1 Retrieve Function OCID



8.2 Update DG_FUNCTIONS

```
ALL {resource.id = 'ocid1.fnfunc.oc1.xxxxxxxxxxxxxxxxxx'}
```

9 Setup Event Service

9.1 Create a rule to detect 'Instance – Launch End'

Create this rule in your ROOT Compartment to receive notifications from any instance in your tenancy.

- Observability & Management > Event Service > Rules

Edit Rule

Display Name
Instance_launch

Description
Describe what the rule does. Example: Sends a notification when backups complete.

Rule Conditions
Limit the events that trigger actions by defining conditions based on event types, attributes, and filter tags. [Learn more](#)

Condition	Service Name	Event Type
Event Type	:	Compute
		Instance - Launch End x

+ Another Condition

Actions
Actions trigger for the specified event conditions. [Learn more](#)

Action Type	Function Compartment	Function Application	Function
Functions	:	IMDS_Check	fn-imds-check

+ Another Action

This function will now be invoked and executed after each instance is launched.

10 Review Function logs:

Launch a new instance and check the function logs

Resources

- Getting started
- Functions
- Configuration
- Signature verification
- Metrics
- Logs**
- Traces

Logs

Category	Status	Log Name
Function Invocation Logs	Active	FBO_Functions_invoke

Explore Log

Sort: Newest | Filter by time: Past 5 minutes | Actions

Number of log events per minute

datetime	type	data.message
Feb 17, 2025, 20:51:54 UTC	functions.application.functioninvoke	Served function invocation request in 5.502 seconds
Feb 17, 2025, 20:51:54 UTC	functions.application.functioninvoke	INFO - FN_IMDS_UPDATE_466: ZZZ88 IMDSv2: True
Feb 17, 2025, 20:51:51 UTC	functions.application.functioninvoke	INFO - FN_IMDS_UPDATE_466: ZZZ88: Updating...
Feb 17, 2025, 20:51:51 UTC	functions.application.functioninvoke	INFO - FN_IMDS_UPDATE_466: Mushop/ZZZ88: com.oraclecloud.compu...

ZZZ88

Start Stop Reboot **Terminate** More actions

Instance information | Shielded instance | Maintenance

General information

Availability domain: AD-1
Fault domain: FD-3
Region: eu-frankfurt-1
OCID: ...svjmwa [Show](#) [Copy](#)
Launched: Mon, Feb 17, 2025, 20:51:14 UTC
Compartment:
Capacity type: On-demand

Instance details

Virtual cloud network: [vcn-20250217-1431](#)
Launch mode: PARAVIRTUALIZED
Instance metadata service: Version 2 only [Edit](#) [i](#)