Oracle Functions (Limited Availability Release)

Set up, creation, and deployment

A. Set up your **tenancy**

0. Subscribe to Phoenix region (LA Release) → **1.** Create groups and users → **2.** Create compartment → **3.** Create VCN and subnets → **4.** Create IAM policies

B. Set up your **client**

1. Set up signing key → **2.** Set up OCI profile → **3.** Install and start Docker → **4.** Install Fn Project CLI → **5.** Create and configure CLI context → **6.** Generate auth token → **7.** Log in to Registry

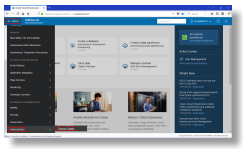
C. Create, deploy, and invoke your **function**

1. Create your first application → **2.** Create your first function → **3.** Deploy your first function → **4.** Invoke your first function → **5.** Next steps

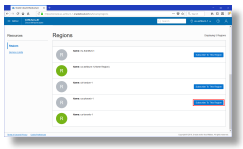
*A. Set up your **tenancy***

0 Subscribe tenancy to Phoenix region (Limited Availability Release only)

To use the Limited Availability Release of Oracle Functions, your tenancy must be subscribed to the Phoenix region. Log in to the Console as a tenancy administrator and under **Governance and Administration**, go to **Administration**, then **Tenancy Details**, and then:

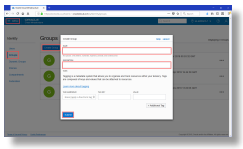


- On the **Regions** page, confirm that your tenancy is already subscribed to the Phoenix region.
- If your tenancy is not yet subscribed to the Phoenix region, click the **Subscribe To This Region** button beside us-phoenix-1.



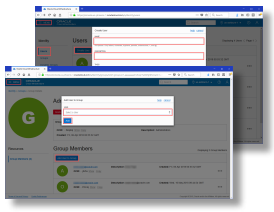
1 Create groups and users

If suitable users and groups don't exist already, log in to the Console as a tenancy administrator and under **Governance and Administration**, go to **Identity** and then:



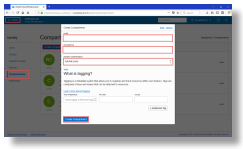
- Create a new group by clicking **Groups** and then **Create Group**.

- Create a new user by clicking **Users** and then **Create User**.
- Add a user to a group by clicking **Groups**, then the name of the group, and then **Add User to Group**.



2 Create compartment

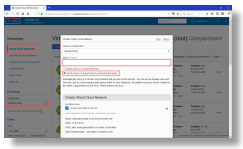
If a suitable compartment doesn't exist already, log in to the Console as a tenancy administrator and under **Governance and Administration**, go to **Identity** and then:



- Click **Compartments** and then **Create Compartment**.

3 Create VCN and subnets

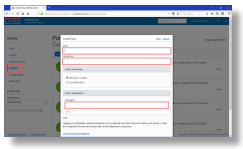
If a suitable VCN doesn't exist already, log in to the Console as a tenancy administrator and under **Core Infrastructure**, go to **Networking** and then:



- Click **Virtual Cloud Networks** and choose a compartment.
- Click **Create Virtual Cloud Network**, enter a name, and select the **Create Virtual Cloud Network Plus Related Resources** option.

4 Create IAM policies

Log in to the Console as a tenancy administrator and under **Governance and Administration**, go to **Identity** and click **Policies** and then:



- Select the tenancy's root compartment, and create two policies to give access to repositories in Oracle Cloud Infrastructure Registry:

  • a policy to give users access to repositories, with one policy statement:

  ```
  Allow group <group-name> to manage repos in tenancy
  ```

  • a policy to give Oracle Functions access, with one policy statement:

  ```
  Allow service FaaS to read repos in tenancy
  ```

- Select the compartment that will own Oracle Functions resources, and create two policies to give access to those resources:

  • a policy to give users access, with three policy statements:

  ```
  Allow group <group-name> to manage functions-family in compartment <compartment-name>
  ```

  ```
  Allow group <group-name> to manage vnics in compartment <compartment-name>
  ```

  ```
  Allow group <group-name> to inspect subnets in compartment <compartment-name>
  ```

  • a policy to give Oracle Functions access, with one policy statement:

  ```
  Allow service FaaS to use virtual-network-family in compartment <compartment-name>
  ```

*B. Set up your **client***

1 Set up signing key

Log in to your development environment as a functions developer and:

- Generate a private key encrypted with a passphrase that you provide by entering:

```
$ openssl genrsa -out ~/.oci/<private-key-file-name>.pem -aes128 2048
```

- Change permissions on the file to ensure that only you can read it.

```
$ chmod go-rwx ~/.oci/<private-key-file-name>.pem
```

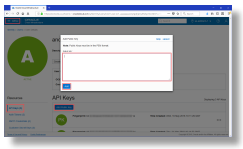
- Generate a public key (encrypted with the same passphrase that you provided when creating the private key, and in the same location as the private key file) by entering:

```
$ openssl rsa -pubout -in ~/.oci/<private-key-file-name>.pem -out ~/.oci/<public-key-file-name>.pem
```

- Copy the contents of the public key file you just created, by entering:

```
$ cat ~/.oci/<public-key-file-name>.pem | pbcopy
```

- Log in to the Console as a functions developer, open the **User** menu ( ) and go to **User Settings**. On the **API Keys** page, click **Add Public Key**. Paste the public key's value into the window and click **Add**. The key is uploaded and its fingerprint is displayed.



2 Set up OCI profile

Log in to your development environment as a functions developer and:

- Open the file ~/.oci/config in a text editor. (If the directory and/or the file don't already exist, create them).
- Add a new profile to the ~.oci/config file as follows:

```
[<profile-name>]
user=<user-ocid>
fingerprint=<public-key-fingerprint>
key_file=<private-key-pem-file>
tenancy=<tenancy-ocid>
region=<region-name>
pass_phrase=<passphrase>
```

- Save and close the file.

3 Install and start Docker

Log in to your development environment as a functions developer and:

- Confirm that Docker is installed by entering:

```
$ docker version
```

- If you see an error message indicating that Docker is not installed, you have to install Docker before proceeding. See the Docker documentation for your platform (for Oracle Linux, see here).
- Assuming Docker is installed, go to the Prerequisites section of the Fn Project home page on GitHub and confirm that the installed version of Docker is at least the minimum version specified there. If not, re-install Docker before proceeding.
- Launch the standard hello-world Docker image as a container to confirm that Docker is running by entering:

```
$ docker run hello-world
```

If you see an error message indicating that Docker is not running, you have to start the Docker daemon before proceeding. See the Docker documentation.

4 Install Fn Project CLI

Log in to your development environment as a functions developer and:

Open the fnproject/cli/README.md file on GitHub and follow the appropriate instructions for installing the Fn Project CLI. As a convenient overview, the instructions are summarized below:

- **MacOS using Homebrew:** Enter:

```
$ brew install fn
```

- **Linux or MacOS:** Enter:

```
$ curl -LSs https://raw.githubusercontent.com/fnproject/cli/master/install | sh
```

- **Linux, MacOS, or Windows:** Download and run the binary from the Fn Project Releases page on GitHub.

Confirm that the CLI has been installed by entering:

```
$ fn version
```

5 Create and configure Fn Project CLI context

Log in to your development environment as a functions developer and:

Create the new Fn Project CLI context by entering:

```
$ fn create context <my-context> --provider oracle
```

Specify that the Fn Project CLI is to use the new context by entering:

```
$ fn use context <my-context>
```

Configure the new context with the OCID of the compartment you want to own deployed functions:

```
$ fn update context oracle.compartment-id <compartment-ocid>
```

Configure the new context with the api-url endpoint to use when calling the Fn Project API by entering:

```
$ fn update context api-url <api-endpoint>
```

For example:

```
$ fn update context api-url https://functions.us-phoenix-1.oraclecloud.com
```

Configure the new context with the address of the Docker registry and repository that you want to use with Oracle Functions by entering:

```
$ fn update context registry <region-code>.ocir.io/<tenancy-name>/<repo-name>
```

For example:

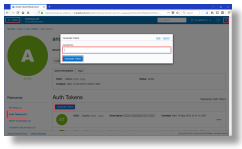
```
$ fn update context registry phx.ocir.io/acme-dev/acme-repo
```

Configure the new context with the name of the profile you've created for use with Oracle Functions by entering:

```
$ fn update context oracle.profile <profile-name>
```

6 Generate auth token

Log in to the Console as a functions developer and:



- Open the **User** menu (👤) and go to **User Settings**. On the **Auth Tokens** page, click **Generate Token**.
- Enter a meaningful description for the auth token in the Generate Token dialog, and click **Generate Token**. The new auth token is displayed.
- Copy the auth token immediately to a secure location from where you can retrieve it later, because you won't see the auth token again in the Console.
- Close the Generate Token dialog.

7 Log in to Registry

Log in to your development environment as a functions developer and:

- Log in to Oracle Cloud Infrastructure Registry by entering:

```
$ docker login <region-code>.ocir.io
```

For example:

```
$ docker login phx.ocir.io
```

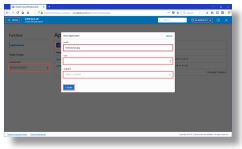
- When prompted, enter the name of the user you will be using with Oracle Functions to create and deploy functions, in the format <tenancy-name>/<username>.
- When prompted for a password, enter the user's Oracle Cloud Infrastructure auth token.

You're now ready to start creating, deploying, and invoking functions.

*C. Create, deploy, and invoke your **function***

1 Create your first application

Log in to the Console as a functions developer, go to the Oracle Functions landing page at https://console.us-phoenix-1.oraclecloud.com/functions and:



- Select the region you intend to use for Oracle Functions (recommended to be the same region as the Docker registry specified in the Fn Project CLI context).
- Select the compartment specified in the Fn Project CLI context.
- Click **Create Application** and specify:

  • `helloworld-app` as the name for the new application. You'll deploy your first function in this application, and specify this application when invoking the function.

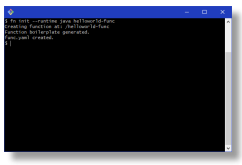
  • The VCN and subnet in which to run the function.
- Click **Create**.

2 Create your first function

Log in to your development environment as a functions developer and:

Create a helloworld java function by entering:

```
$ fn init --runtime java helloworld-func
```



A directory called helloworld-func is created, containing:

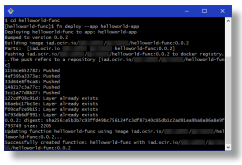
• a function definition file called func.yaml

• a /src directory containing source files and directories for the helloworld function

• a Maven configuration file called pom.xml that specifies the dependencies required to compile the function

3 Deploy your first function

Log in to your development environment as a functions developer and:

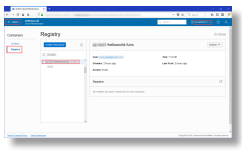


Change directory to the helloworld-func directory created in the previous step:

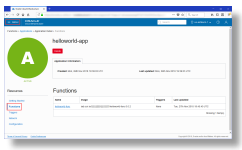
```
$ cd helloworld-func
```

Enter the following single Fn Project command to build the function and its dependencies as a Docker image called helloworld-func, push the image to the specified Docker registry, and deploy the function to Oracle Functions in the helloworld-app application that you created earlier:

```
$ fn deploy --app helloworld-app
```



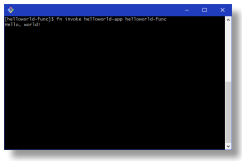
(Optional) Confirm that the helloworld-func image has been pushed to Oracle Cloud Infrastructure Registry by logging in to the Console as a functions developer. Under **Solutions, Platform and Edge**, go to **Developer Services** and click **Registry**. Choose the registry's region, then click the name of the repository you specified in the Fn Project CLI context to see the helloworld-func function within it.



(Optional) Confirm that the function has been deployed to Oracle Functions by logging in to the Console as a functions developer. Go to the Oracle Functions landing page at https://console.us-phoenix-1.oraclecloud.com/functions. Select the compartment you specified in the Fn Project CLI context, then click the helloworld-app on the Applications page to see that the helloworld-func function has been deployed to Oracle Functions.

4 Invoke your first function

Log in to your development environment as a functions developer and:



Invoke the helloworld-func function in the helloworld-app that you created earlier by entering:

```
$ fn invoke helloworld-app helloworld-func
```

The 'Hello World !' message is displayed.

Congratulations! You've just created, deployed, and invoked your first function using Oracle Functions!

5 Next steps

Now that you've created, deployed, and invoked a function, read the Oracle Functions User Guide to find out how to:

• export function logs by configuring a syslog URL (see "Exporting Function Logs")

• pass parameters when invoking a function (see "Using Fn Project CLI Commands to Invoke Functions")

• invoke a function using oci-curl (see "Sending a Signed Request to a Function's Invoke Endpoint (using oci-curl)")

• invoke a function using SDKs (coming soon!)

 **You're done!**

Find out more about ***Oracle Cloud Infrastructure and Oracle Functions***

Product Information

See: User Guide (PDF), cloud.oracle.com, docs.cloud.oracle.com, Fn Project

Attend Oracle Cloud Events

See: events.oracle.com, blogs.oracle.com

Join the Community

See: Customer Community, Forums

Follow Oracle Cloud

Facebook Twitter YouTube