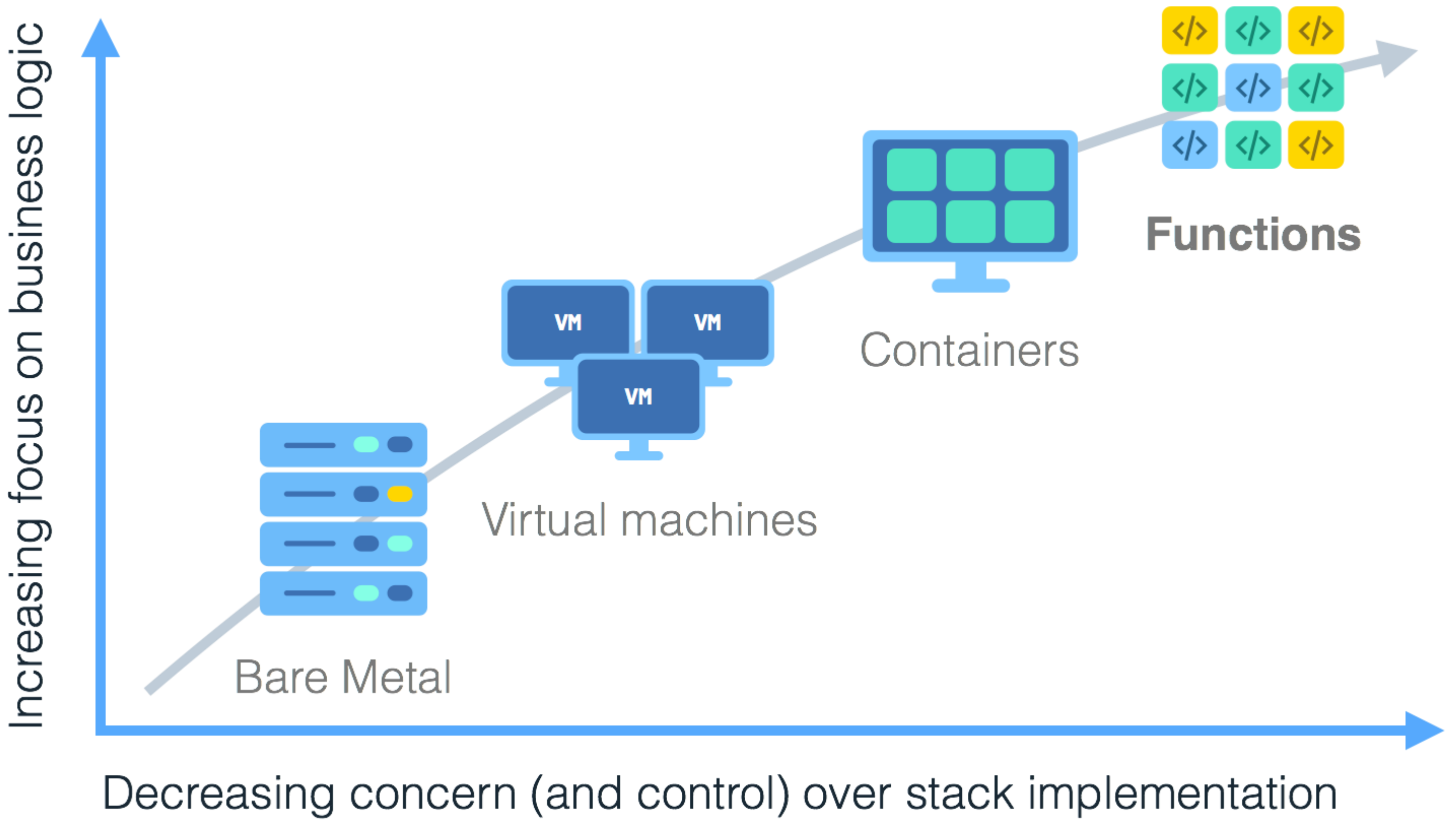
IBM Cloud

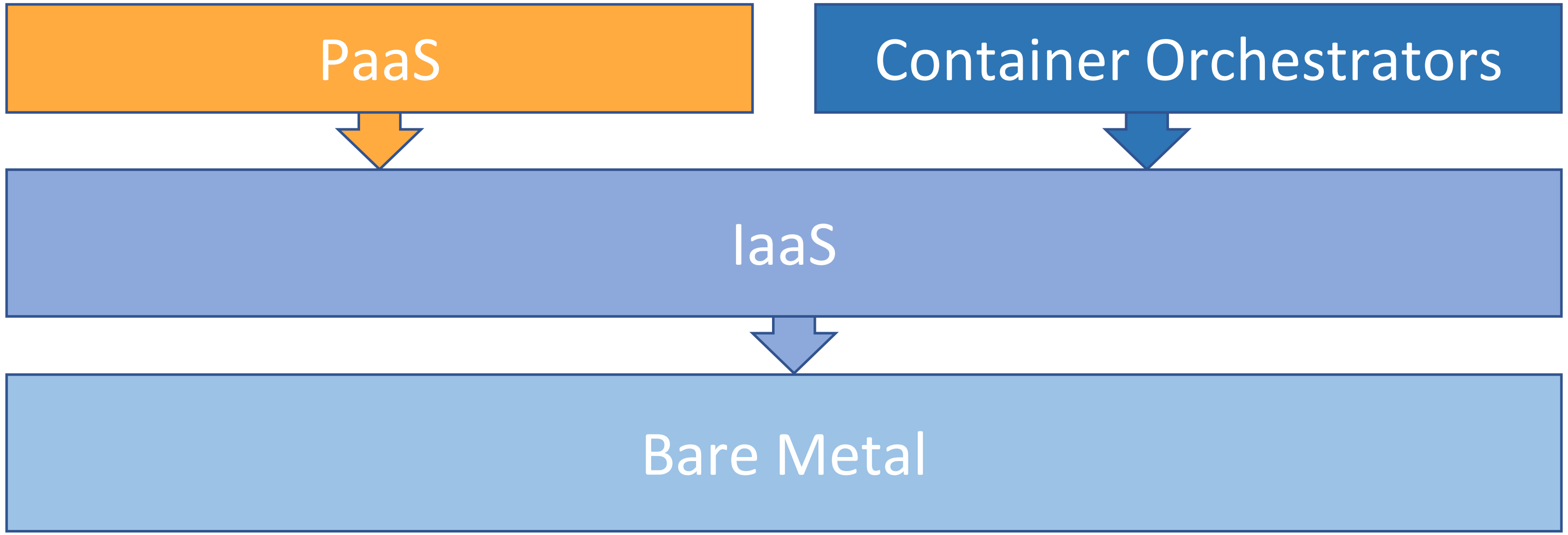
IBM Cloud Functions - Introduction

Josep Sampé

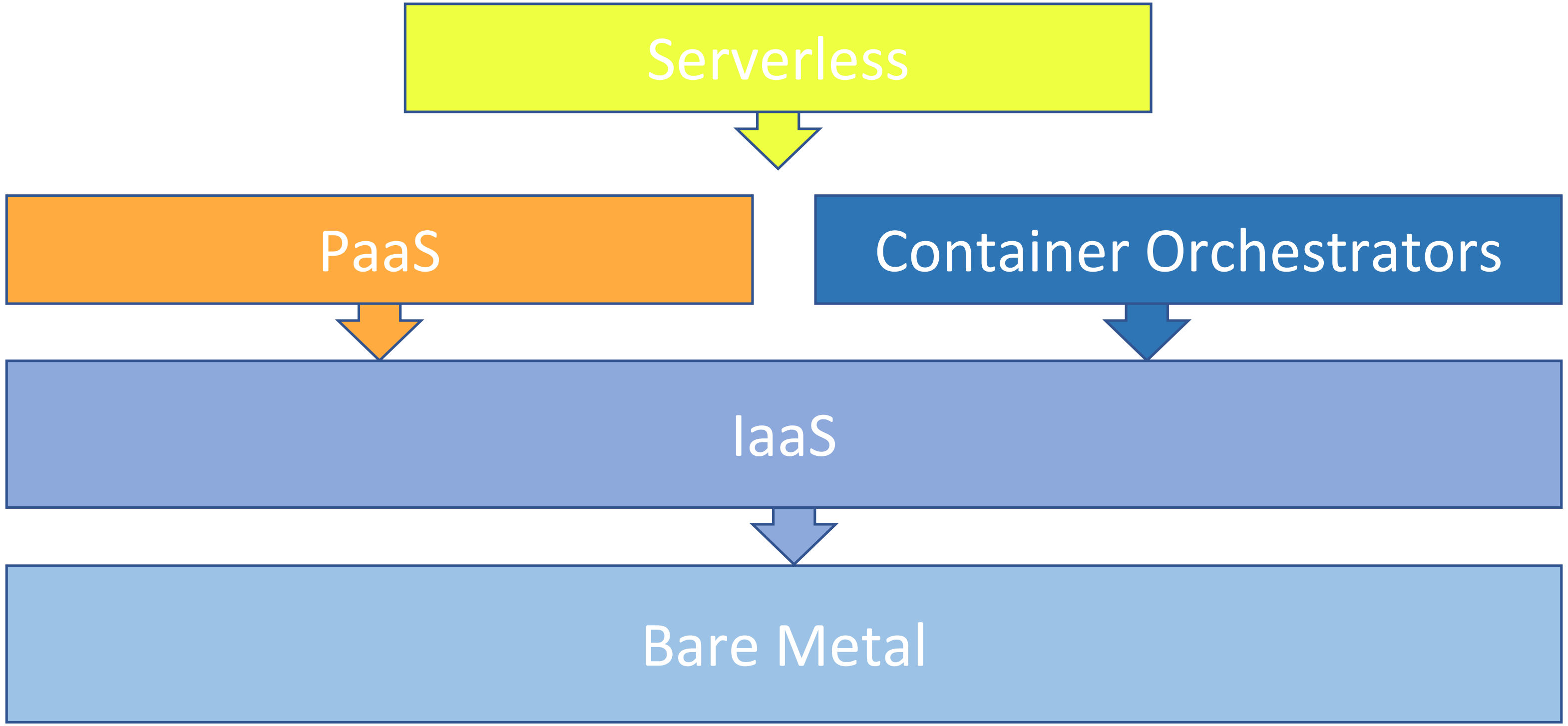
Evolution Of Serverless


Increasing focus on business logic
Functions
VM
VM
VM
Containers
Virtual machines
Bare Metal
Decreasing concern (and control) over stack implementation

Evolution Of Serverless



Evolution Of Serverless



What is Serverless?

a cloud-native platform

for

short-running, stateless computation

and

event-driven applications

which

scales up and down instantly and automatically

and

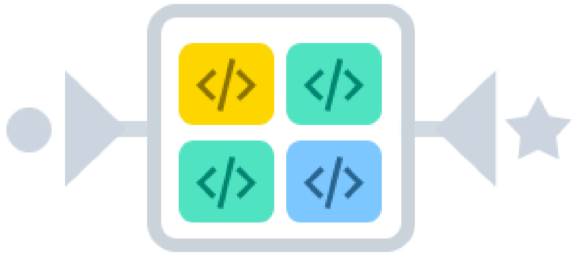
charges for actual usage at a millisecond granularity

Server-less means no servers?
Or worry-less about servers?

Runs code **only** on-demand on a per-request basis

Serverless deployment & operations model

No servers

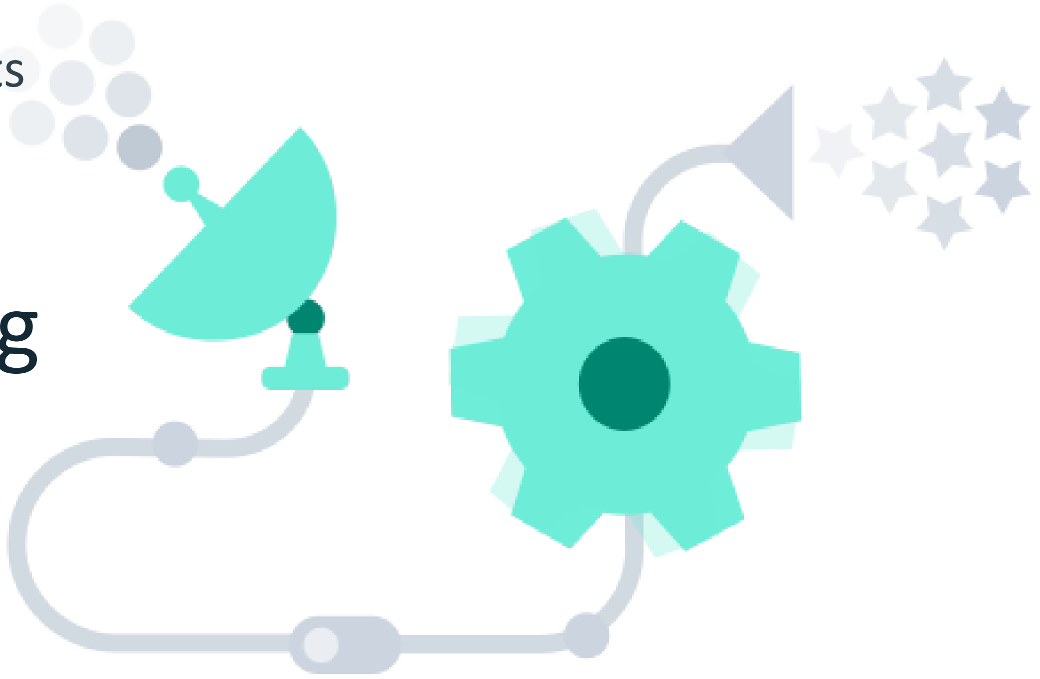


Just code

What triggers code execution?

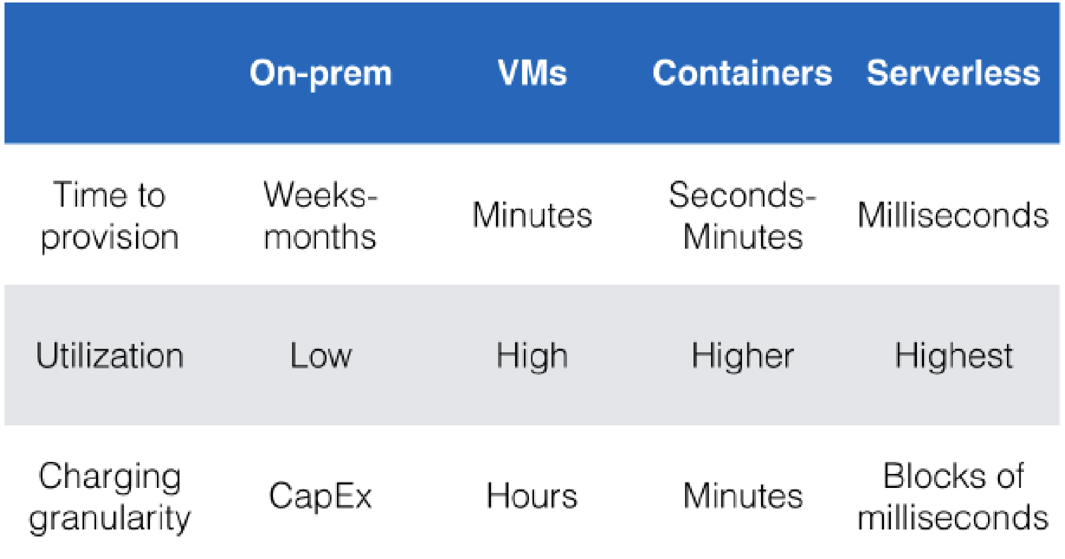
Runs code **in response** to events

Event-programming model



Why is Serverless attractive?

- Making app development & ops dramatically faster, cheaper, easier
- Drives infrastructure cost savings



	On-prem	VMs	Containers	Serverless
Time to provision	Weeks-months	Minutes	Seconds-Minutes	Milliseconds
Utilization	Low	High	Higher	Highest
Charging granularity	CapEx	Hours	Minutes	Blocks of milliseconds

Source: Jason McGee, IBM; Serverless Conference 2017.

What is Serverless good for?

Serverless is good for

short-running
stateless
event-driven

- Microservices
- Mobile Backends
- Bots, ML Inferencing
- IoT
- Modest Stream Processing
- Service integration

Serverless is not good for

long-running
stateful
number crunching

- Databases
- Deep Learning Training
- Heavy-Duty Stream Analytics
- Numerical Simulation
- Video Streaming

Current Platforms for Serverless

APACHE OpenWhisk

AWS
Lambda

OpenLambda

Iron.io

Azure Functions

fission

Kubernetes

IBM Cloud Functions

Red-Hat

Google Functions

fn

Account Setup

Account Setup

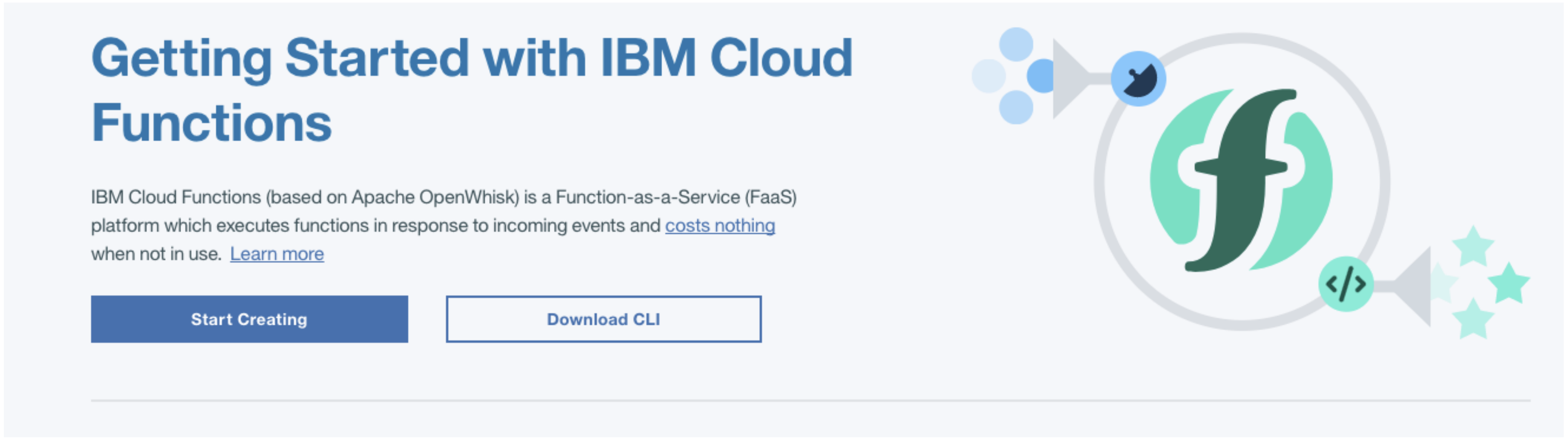
- Log-in with your IBM Cloud account: create one if you do not yet have one by clicking the sign-up link or by directly navigating to https://cloud.ibm.com/registration to get IBM Cloud account

- To use IBM Cloud Functions proceed as follows: open a browser window and navigate to https://cloud.ibm.com/openwhisk

- Click “Start Creating” to create cloud functions directly from browser

- Click the Download command line tools for your operating system: https://cloud.ibm.com/openwhisk/learn/cli
  - Follow steps 1 & 2 & 3 & 4, i.e. download the CLI for your particular platform and configure it by specifying your namespace and authorization key.

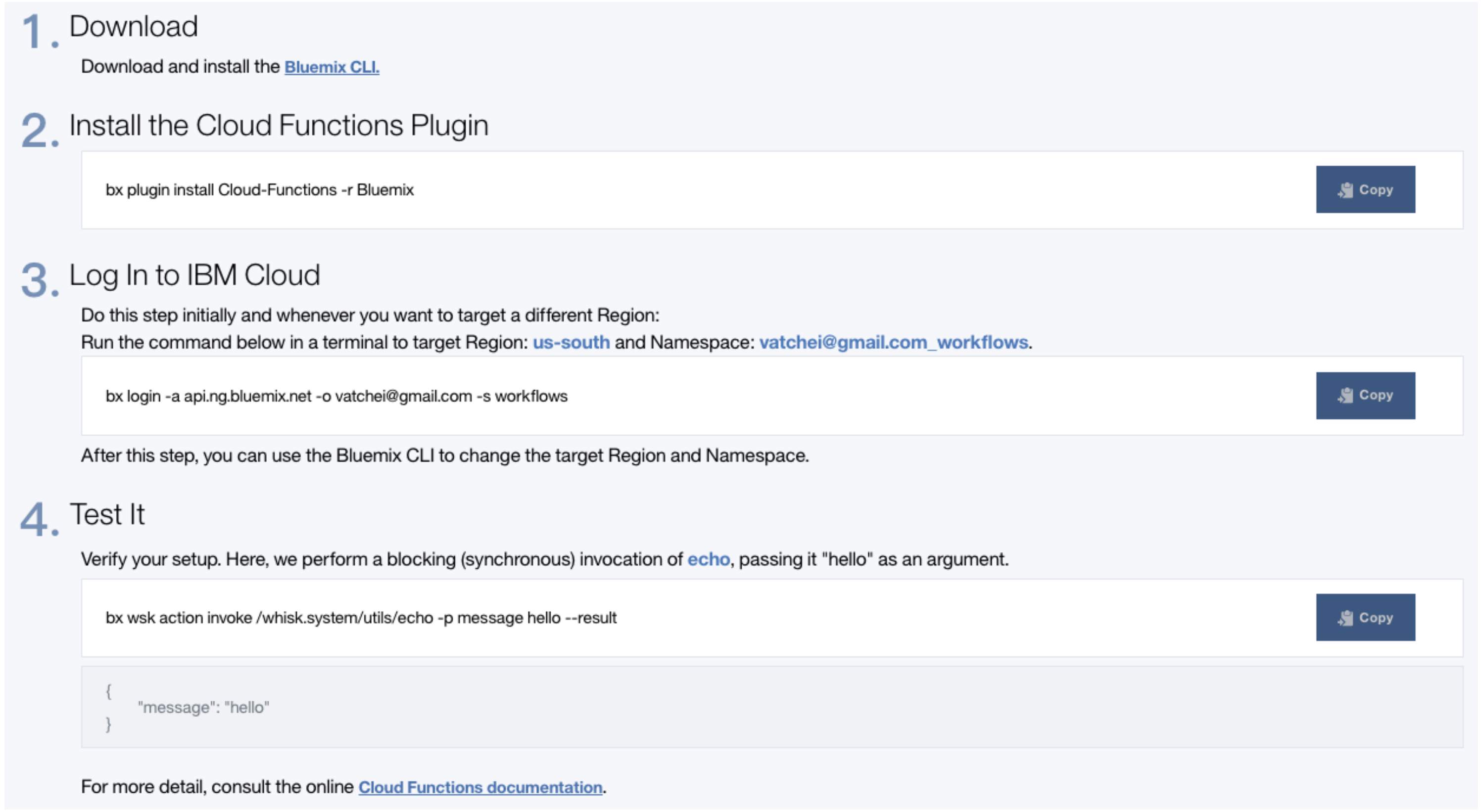
IBM Cloud Functions screenshot



Getting Started with IBM Cloud Functions

IBM Cloud Functions (based on Apache OpenWhisk) is a Function-as-a-Service (FaaS) platform which executes functions in response to incoming events and costs nothing when not in use. Learn more

IBM Cloud Functions client



1. Download

Download and install the Bluemix CLI.

2. Install the Cloud Functions Plugin

```
bx plugin install Cloud-Functions -r Bluemix
```

3. Log In to IBM Cloud

Do this step initially and whenever you want to target a different Region:
Run the command below in a terminal to target Region: us-south and Namespace: vatchei@gmail.com_workflows.

```
bx login -a api.ng.bluemix.net -o vatchei@gmail.com -s workflows
```

After this step, you can use the Bluemix CLI to change the target Region and Namespace.

4. Test It

Verify your setup. Here, we perform a blocking (synchronous) invocation of echo, passing it "hello" as an argument.

```
bx wsk action invoke /whisk.system/utils/echo -p message hello --result
```

```
{
    "message": "hello"
}
```

For more detail, consult the online Cloud Functions documentation.

Using **ibmcloud** command line tool

- Login to IBM Cloud

```
> ibmcloud login -a cloud.ibm.com
```

- Install the Cloud Functions pluguin:

```
> ibmcloud plugin install cloud-functions
```

- Invoke the test action:

```
> ibmcloud fn action invoke /whisk.system/utils/echo
-p message hello --result
```

Python Actions

Task: Creating and invoking Python actions

- An action can be a simple Python function that accepts and returns a JSON object.
- Create a file called hello.py

```
def main(args):
    name = args.get("name", "stranger")
    greeting = "Hello " + name + "!"
    print(greeting)
    return {"greeting": greeting}
```

- Create an IBM Cloud action called hello

```
ibmcloud fn action create hello hello.py
```

Task: Creating and invoking Python actions

- List the actions you created

```
ibmcloud fn action list
```

- To run an action use the wsk action invoke command.

```
ibmcloud fn action invoke --blocking hello
```

- You can retrieve the list of activations at any time

```
ibmcloud fn activation list
```

- Enter the invocation ID shown, for example:

```
ibmcloud fn activation get dde9212e686f413bb90f22e79e12df74
```

- You can delete an action

```
ibmcloud fn action delete hello
```

Task: Passing parameters to actions

- Create the action

```
ibmcloud fn action create hello hello.py
```

- You can pass named parameters as JSON payload or via the CLI

```
 ibmcloud fn action invoke --result helloPython --param name World

{
    "message": "Hello, World"
}
```

Task: Using actions to call an external API

- Develop a **COS Reader** action that reads a file from COS and returns its content to the client:

1. Upload a file to COS by using the **cos_backend.py** module.
2. Create the **__main__.py** file and add the *main(args)* method.
3. Pass through the “args” the name of the file uploaded to COS.
4. Import de **cos_backend.py** module, and retrieve the previous uploaded file.
    - data = cos.get_object(bucket_name, file_name)
5. Return the content of the file through the action.
    - return {'file_content': data}

Task: Using actions to call an external API

- Run the following commands to create the action and invoke it

```
zip -r cosreader.zip __main__.py cos_backend.py

ibmcloud fn action create cosreader --kind python:3.6
cosreader.zip

ibmcloud fn action invoke --blocking --result
cosreader --param filename “data1.txt”
```