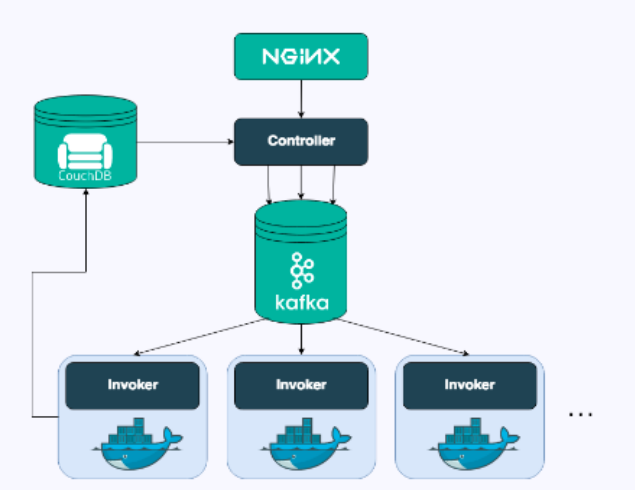
Assignment2:

Install Openwhisk on your local machine using the instruction below. Then implement a hello world function using node.js.

A Quick Overview on OpenWhisk:

1-Architecture:



The above diagram depicts the high-level architecture of OpenWhisk. From Nginx to Kafka to Docker, multiple technologies are powering Apache OpenWhisk - Open Source Serverless Cloud Platform.

2-Deployment:

OpenWhisk supports different installation methods for core OpenWhisk platform components. You can choose one of the following deployment methods based on your platform:

- **Docker Compose:** The easiest way to start deploying OpenWhisk is to get Docker installed on Mac, Windows or Linux. This does not give you a production deployment but gives you enough of the pieces to start writing functions and seeing them executing.
- **Kubernetes:** OpenWhisk can be installed on Minikube, or on a managed Kubernetes cluster provisioned from a public cloud provider, or on a Kubernetes cluster you manage yourself.
- **Mesos**
- **Openshift**
- **Ansible:** Deploying OpenWhisk using Ansible is generally done for creating a development environment. Most of the OpenWhisk tools follow this method of deployment in their CI/CD (Travis) pipeline. The OpenWhisk playbooks are structured such that it allows cleaning, deploying, or re-deploying a single component as well as the entire OpenWhisk stack.
- **Vagrant**

There are many different deployment methods available but it is recommended to Deploy OpenWhisk to a Kubernetes Cluster as it's the most easy and quickest way of getting OpenWhisk deployed.

3-Running OpenWhisk Using Docker Compose:

In order to Run OpenWhisk using Docker Compose, First of All we have to install docker on our system.

Now Consider we have a Debian9 OS. The first step is to install Docker

3-1- Docker Installation

```
sudo apt-get remove docker docker-engine docker.io
sudo apt-get update
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg2 \
    software-properties-common
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
sudo apt-key fingerprint 0EBFCD88
sudo add-apt-repository \
  "deb [arch=amd64] https://download.docker.com/linux/debian \
  $(lsb_release -cs) \
  stable"
sudo apt-get update
sudo apt-get install docker-ce
sudo curl -L "https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$
(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

Above Installation Instruction is based on the Official Document.

3-2- Running OpenWhisk using Docker Compose

These Instructions are based on the official Document that Openwhisk Suggested.

```
git clone https://github.com/apache/incubator-openwhisk-devtools.git
cd incubator-openwhisk-devtools/docker-compose
make quick-start
```

Using the above command openwhisk is ready to use, a simple application that can be executed is hello world. As an example it has been written in Javascript:

```
function main(params) {var name = params.name || "World"; return { payload:  "Hello, " + name
+ "!" }; }
```

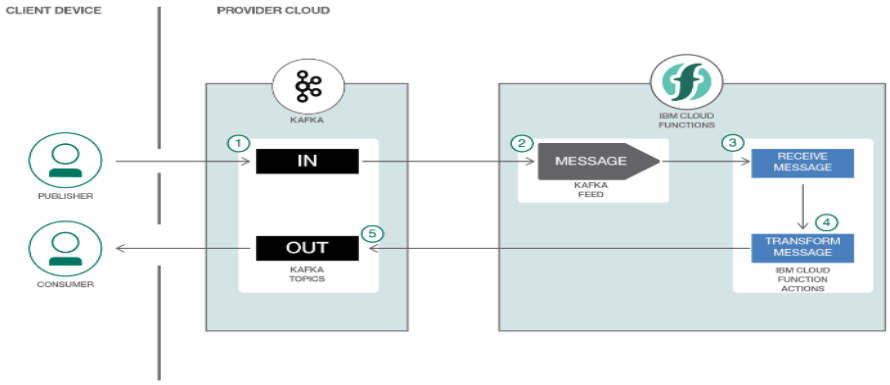
If you run “ make hello-world” openwhisk execute it and print out :

```
{ "payload": "Hello, World!" }
```

4- Openwhisk and Data Streams:
There is an Official Project by IBM that shows how OpenWhisk is working with Data Streams.
This project shows how serverless, event-driven architectures execute code in response to messages or to handle streams of data records.

The application demonstrates two IBM Cloud Functions (based on Apache OpenWhisk) that read and write messages with IBM Message Hub (based on Apache Kafka). The use case demonstrates how actions work with data services and execute logic in response to message events.

One function, or action, is triggered by message streams of one or more data records. These records are piped to another action in a sequence (a way to link actions declaratively in a chain). The second action aggregates the message and posts a transformed summary message to another topic.



Note that this project is using IBM Cloud Functions and Kafka, We can use OpenWhisk instead of Cloud Functions and also create a local instance of kafka in order to reproduce this example.

5- Dark Vision:

Dark Vision is a technology demonstration leveraging Cloud Functions and Watson services.

Think about all the videos individuals and companies (Media and Entertainment) accumulate every year. How can you keep track of what's inside of them so you can quickly search and find what you're looking for? *Show me all the videos that have Arc De Triomphe in it.* or *Show me the all the videos that talk about peaches.*

What if we used artificial intelligence to process these videos to tell us which video has what we're looking for without us having to watch all of them.

Dark Vision is an application that processes videos to discover what's inside of them. By analyzing individual frames and audio from videos with IBM Watson Visual Recognition and Natural Language Understanding, Dark Vision builds a summary with a set of tags, famous people or landmarks detected in the video. Use this summary to enhance video search and categorization.

5-1-Steps:

- Extracting Frames and Audios from a Video
- Processing Frames and standalone Images
- Processing Audio
    - Extract the Transcript
    - Analyze the Transcript

We can use Openwhisk instead of Cloud Function in this project. Note that we have to provide the Infrastructure.

6- Further Reading

This web Article explains more about 5 Emerging Use Cases of the IBM OpenWhisk Serverless Platform. As it Describes, There is 5 use cases as below:

- Mobile Backend Development
- Data Processing
- Cognitive Processing
- Streaming Analytics
- Internet of thing