Serverless computing hands on lab

This tutorial will give you hands on experience with the Apache OpenWhisk serverless computing platform

- These slides are available at http://goo.gl/iQuK6m
- The code in this tutorial is at https://github.com/serverless-workshop/tutorial

Part 0

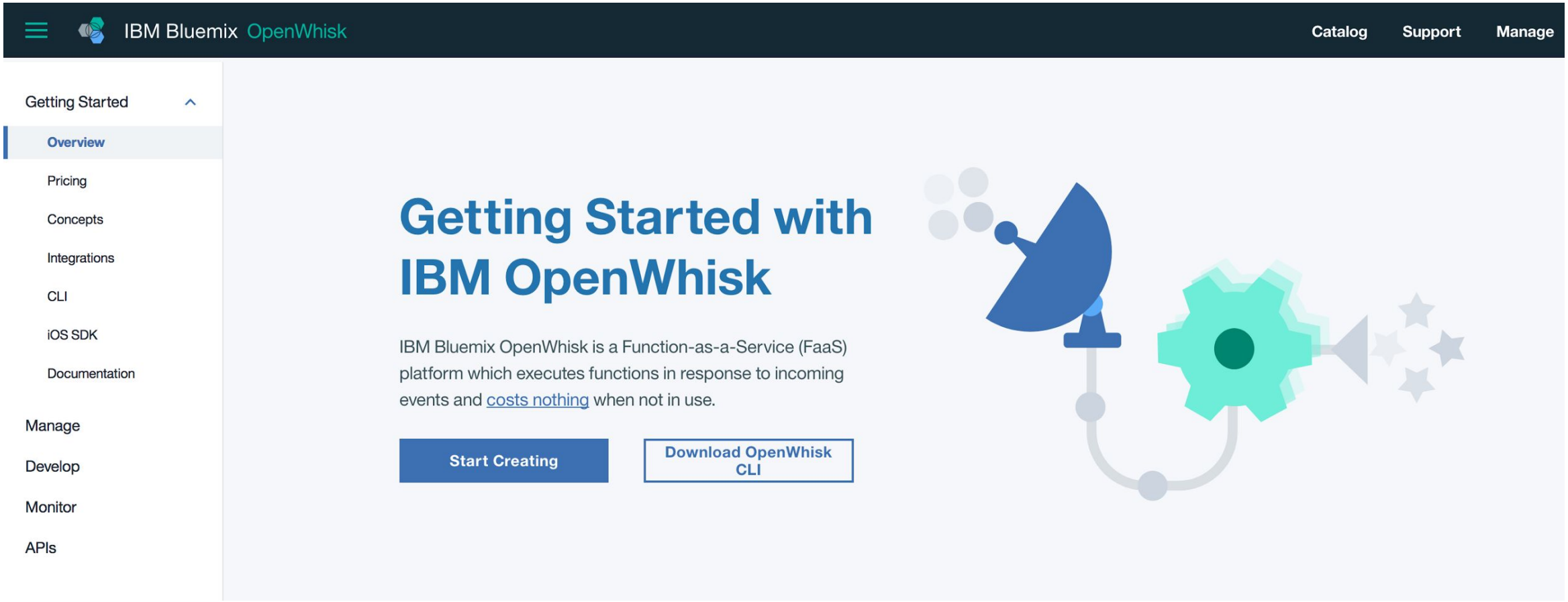
Account setup

slides are available at http://goo.gl/iQuK6m

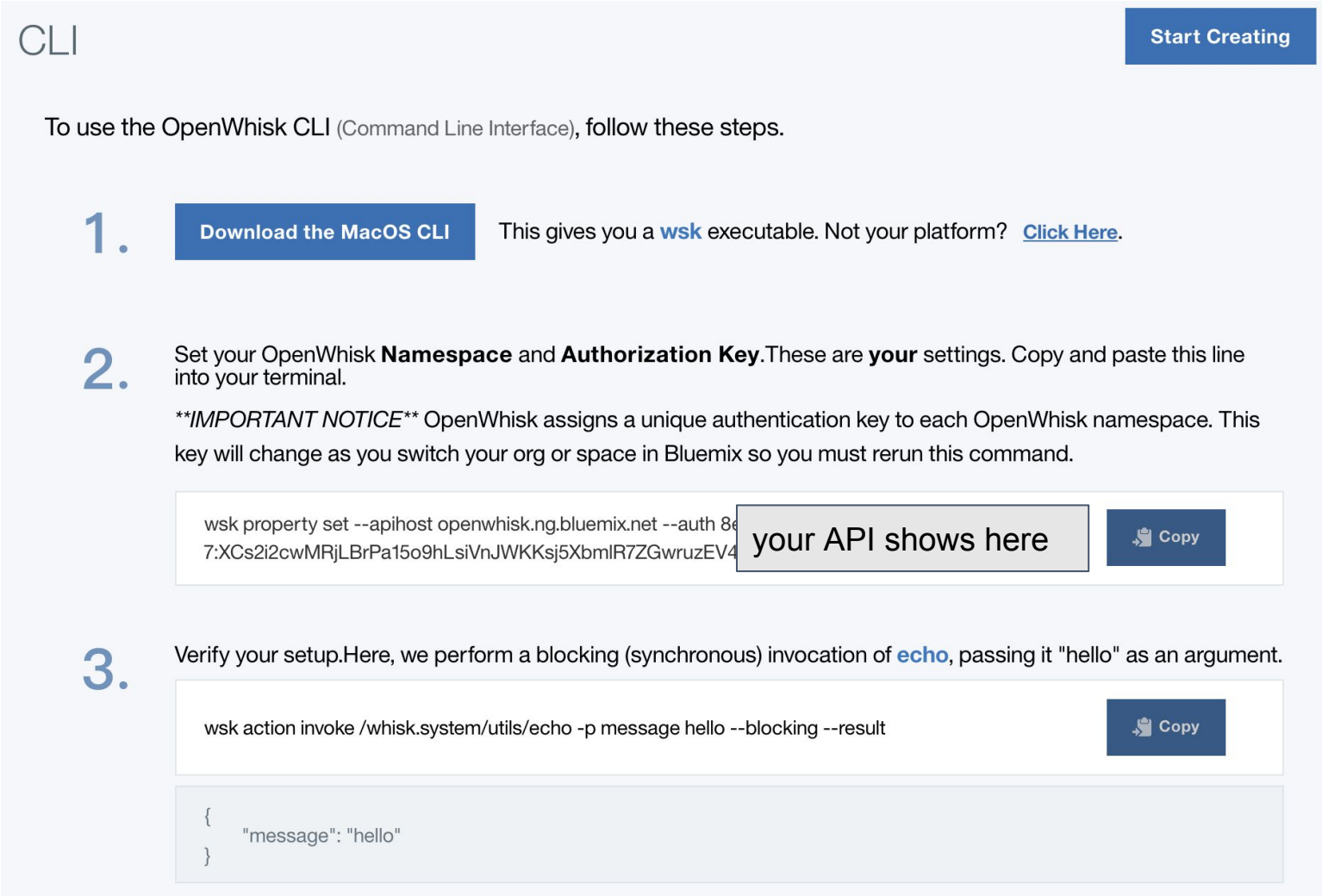
Tutorial Setup

- To use OpenWhisk proceed as follows: open a browser window and navigate to https://console.ng.bluemix.net/openwhisk/
- Log-in with your Bluemix account Create one if you do not yet have one by clicking the sign-up link or by directly navigating to https://console.ng.bluemix.net/registration/
- Click the Download OpenWhisk CLI https://console.ng.bluemix.net/openwhisk/learn/cli
  Direct download link https://openwhisk.ng.bluemix.net/cli/go/download/
- Follow steps 1 & 2 (you do not need to perform step 3), i.e. download the CLI for your particular platform and configure it by specifying your namespace and authorization key

Bluemix screenshot



Bluemix screenshot



Confirm all is working:

Make sure to run wsk command that set your API key (Step 2)

wsk property set --apihost openwhisk.ng.bluemix.net --auth YOUR-KEY-HERE

and test it:

wsk action invoke /whisk.system/utils/echo -p message ICDCS --blocking --result

output:

{ "message": "ICDCS" }

Part 1

OpenWhisk actions, triggers, rules

Creating and invoking JavaScript actions

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

```
function main()
{
    return { message: "Hello world" };
}
```

- Create an OpenWhisk action called hello

```
wsk action create hello hello.js
```

Creating and invoking JavaScript actions

- List the actions you created

  `wsk action list`

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

  `wsk action invoke --blocking hello`

- You can retrieve the list of activations at any time

  `wsk activation list`

- Enter the invocation ID shown, for example:

  `wsk activation get` *`dde9212e686f413bb90f22e79e12df74`*

- You can delete an action

  `wsk action delete hello`

Passing parameters to actions

- Change (and save) your hello action as follows

```
function main(msg) {
    return { message: "Hello, " + msg.name + " from " + msg.place };
}
```

- Create the action

```
wsk action create hello hellowithparams.js
```

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

```
wsk action invoke -b hello -p name "Bernie" -p place "Vermont" --result
{
    "message": "Hello, Bernie from Vermont"
}
```

Using actions to call an external API

```
var request = require("request");

function main(msg) {
    var location = msg.location || "Vermont";
    var url = "https://query.yahooapis.com/v1/public/yql?q=select item.condition from weather.forecast where woeid in (select
woeid from geo.places(1) where text='" + location + "')&format=json";

    return new Promise(function(resolve, reject) {
        request.get(url, function(error, response, body) {
            if (error) {
                reject(error);
            }
            else {
                var condition = JSON.parse(body).query.results.channel.item.condition;
                var text = condition.text;
                var temperature = condition.temp;
                var output = "It is " + temperature + " degrees in " + location + " and " + text;

                resolve({msg: output});
            }
        });
    });
}
```

Using actions to call an external API

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

```
wsk action create yahooWeather weather.js
wsk action invoke --blocking --result yahooWeather
--param location "Brooklyn, NY"
```

wsk action invoke --blocking --result yahooWeather --param location "Atlanta, GA"

```
{
    "msg": "It is 75 degrees in Atlanta, GA and Cloudy"
}
```

Triggers and Rules

- Let’s create a trigger to send user location updates:

```
wsk trigger create locationUpdate
wsk trigger list
```

- So far we have only created a named channel to which events can be fired. Now lets fire the trigger.

```
wsk trigger fire locationUpdate -p name "Donald" -p place "Washington, D.C"
```

Triggers and Rules

- *Rules* are used to associate a trigger with an action

```
wsk rule create myRule locationUpdate hello
```

```
wsk trigger fire locationUpdate -p name "Donald" -p place "Washington, D.C"
```

- Check whether the action was really invoked

```
wsk activation list hello
```

- Enter the top invocation ID, for example:

```
wsk activation result 12ca88d404ca456eb2e76357c765ccdb
```

Part 2

Slack integration

Get Access to tutorial slack server

- Join the Slack team: https://future-compute.slack.com
    - Send an email to wosc1717 at gmail.com with your email address and we will send you an invite to the slack server
- Join the #tutorial channel in the Slack team
    - Click on CHANNELS or from slack server run /join tutorial

Post to Slack from OpenWhisk

- Create an incoming webhook integration
  - Documentation https://api.slack.com/incoming-webhooks
  - Go to your slack channel and open preferences
  - Configure it to send messages to the #tutorial channel
  - Record the Webhook URL
    - It should look something like https://hooks.slack.com/services/T8NGB8FEA/B8NHT9VQD/1cskpNAu8VjSC
- Send a message from an OpenWhisk action to your Slack channel
  - wsk action invoke /whisk.system/slack/post \
    -p url https://hooks.slack.com/services/T8NGB8FEA/B8NHT9VQD/1cskpNAu8VjSC \
    -p channel *tutorial* -p text "hello from YOUR_NAME whisk action"
  - Note: change tutorial to you slack channel
- You should see the message in Slack

Invoke an OpenWhisk action from Slack

- Test you can run an existing OpenWhisk Web action
  - curl -X POST -H 'Content-Type: application/json' -d '{"text":"foo"}' 'https://openwhisk.ng.bluemix.net/api/v1/web/vmuthus%40us.ibm.com_dev/default/timenow.json'

  - This should return something like

```
{
    "text": "The time is Mon Jun 05 2017 02:58:49 GMT+0000 (UTC)"
}
```

Invoke an OpenWhisk action from Slack

- Create an Outgoing Webhook integration
  - Documentation https://api.slack.com/custom-integrations/outgoing-webhooks
  - Configure the Slack channel to listen on (e.g., #tutorial) in channel preferences
  - Configure a trigger word (e.g., your name)
  - Configure the URL: https://openwhisk.ng.bluemix.net/api/v1/web/vmuthus%40us.ibm.com_dev/default/timenow.json
- Type something in the Slack channel you configured above with the trigger word. You should see the current time in Slack.
- You've just created a Slack chatbot backed by a serverless backend!

Use a custom Web action

- Now create your own Web action that returns the time
  - wsk action create *mywebaction* timenow.js --web true
  - (The timenow.js file is in the git repo)
- Test that you can invoke it:
  - curl 'https://openwhisk.ng.bluemix.net/api/v1/web/ORG/default/mywebaction.json'
  - Replace the value of *ORG* based on the fully qualified name of your action (do a "wsk list" to see this) - replace '@'with '%40' to get
    - Example: curl https://openwhisk.ng.bluemix.net/api/v1/web/aslom%40us.ibm.com_dev02/default/mywebaction.json
- Update your Slack outgoing Webhook integration with the URL to your action

Part 3

Invoke external services from chatbot

Make your chatbot do something interesting

- Modify your web action code to do something more interesting than return the time

- Here are some ideas
  - Return a random joke by calling this api: https://api.chucknorris.io/jokes/random
  - Reply back with a translated string using the Watson Language Translation API: https://www.ibm.com/watson/developercloud/language-translator.html
  - Return the weather forecast based on a user-specified location using the Yahoo Weather API: https://developer.yahoo.com/weather/
  - Parse the message and return an appropriate response. Can you beat the Turing test?!

Choose your own adventure

- Build a weather Chatbot with OpenWhisk
  - https://github.com/IBM-Bluemix/openwhisk-workshops/tree/master/bootcamp

- Build a video sharing website with AWS Lambda
  - https://github.com/ACloudGuru/serverless-workshop

Come to workshop

- Workshop afternoon with papers and panel discussion

http://www.serverlesscomputing.org/wosc17/

Room: Columbia

First part of tutorial:

https://goo.gl/M8tV8R