

Functions as a Service - ForwardJS

IBM Code

Upkar Lidder
Developer Advocate, IBM

- > ulidder@us.ibm.com
- > [@litterupk](#)
- > blog.upkarlitter.com

Agenda

9:30	10:30	Introduction to Serverless
10:30	10:45	Break (set up CLI)
10:45	12:00	Lab 1 – Hello world, params, logs, async Ex 1
12:00	13:00	Lunch
13:00	13:30	Lab 2 – sequences, NPM bundle and Packages Ex 1.2 ,1.3
13:30	14:00	Lab 3 – Triggers and Rules
14:00	15:45	Ex 3 – Building registration application
15:45	16:00	Break
16:00	16:30	Web Actions API Gateway Resources and What's Next

Workshop and Slides

<https://github.com/lidderupk/openwhisk-workshops>

Actions and General CLI Introduction

- ibmcloud or ic
- ic target -cf
- ic resource --help
- Wskprops
- ic fn list

- ~/.bluemix – delete and restart if unable to bind to a service (should not have to do this)

- ibmcloud UI introduction

Actions and General CLI Introduction



- List actions
- Create helloworld action
- Invoke helloworld action
- Invoke with blocking
- Activation poll, list, limit
- Update Actions
- Use another function as entry point
- Promises for async actions
- Show on IBMCloud UI



Exercise – currency converter from bitcoin to XXX

Lunch Break

Sequence, NPM, Packages

- Create a sequence of actions (split, reverse, join)
 - Activation for sequence
 - NPM – loremipsum (how to use external packages)
 - List packages
 - List packages in /whisk.system namespace
 - Get details for a package
 - Get summary for a package
 - Bind /whisk.system/samples to local package
 - Bind can be used to assign default parameters
 - Create custom package and share with ibmcloud
 - Show on IBMCloud UI
-  Add authentication to sequence of actions created before
-  Upload zip package with your favorite npm module

Rules and Triggers

- Build an action that says "Hello <name> from <place>"
- Build a trigger
- Build a rule that binds the above trigger and action
- Bind default place to the trigger
- Show activation logs
- OpenWhisk NPM package to invoke trigger programmatically
- Show on IBMCloud UI



Build an app that welcomes a user to your site by sending a welcome email and sms

Registration Application



Contract Registration Page

Hiya,

Enter your email and/or phone number and see the OpenWhisk magic

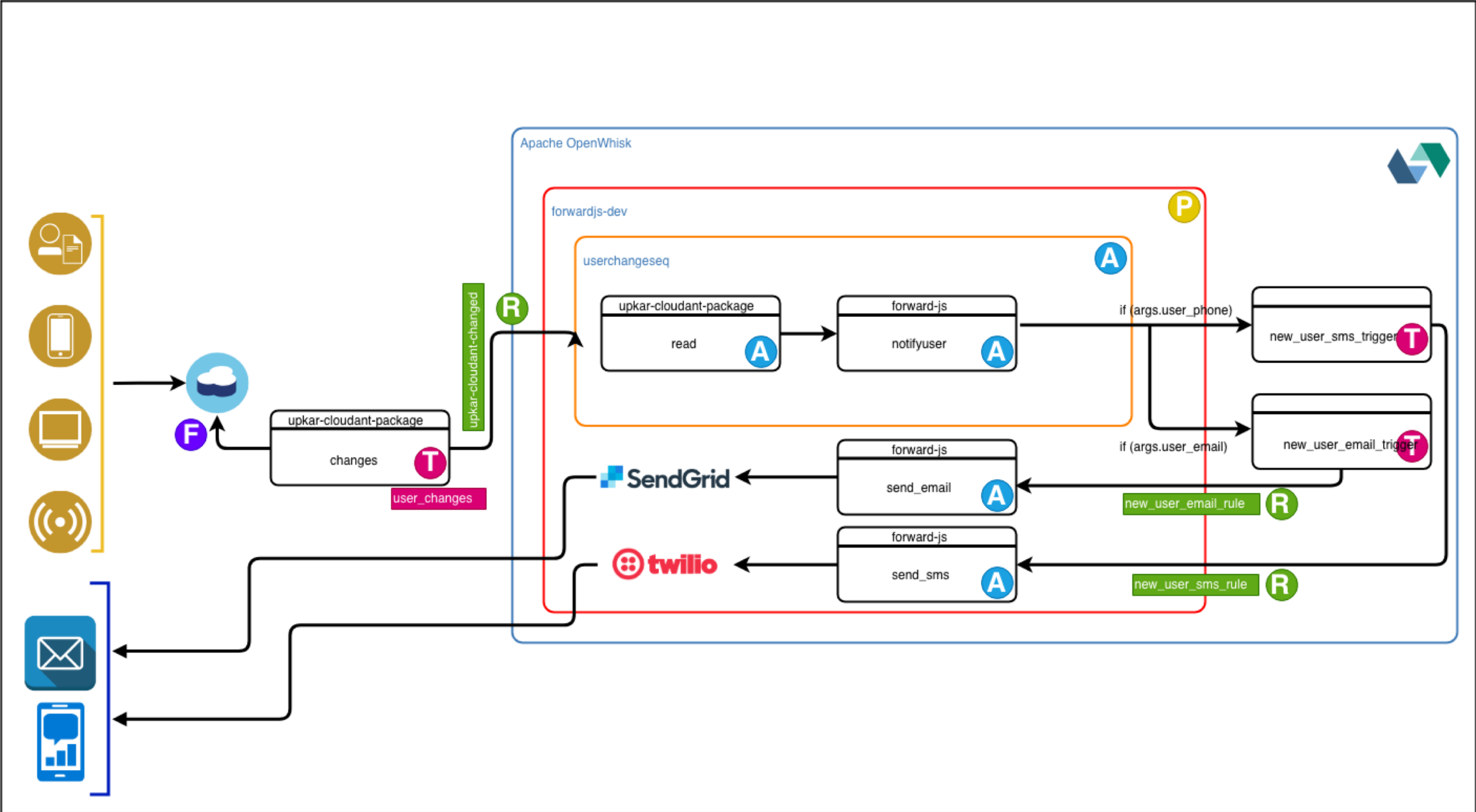
upkar.ibm.watson@gmail.com

+14155701199

Register

Made with [Glitch!](#)

Registration Application



Registration Application

- Create IBM Cloud account and download ibmcloud cli and the fn plugin
- Create cloudbant service through the UI or the cli
- Create an API key for cloudbant if one if not created already
- Launch the cloudbant service and create a new database called “testdb”. You can leave it blank for now.
- Bind the /whisk.system package to upkar-cloudbant-package <use different name>
- Bind the cloudbant service instance to this package
 - ***ic fn service bind cloudbantnosqlldb upkar-cloudbant-package --instance Cloudbant-rf***

Registration Application

- Create forwardjs-dev package
- Create sendmail action within the forwardjs-dev package. This send an email using the Sendgrid service
- Use the default parameters to add the Sendgrid secret token to this action.
- Test the sendemail action with the CLI
- Create the sendsms action within the forwardjs-dev package. This sends a sms using the Twilio service
- Use the default parameters to add the Twilio secret token, accountSID and the trial number to this action.
- Test the sendsms action with the CLI

Registration Application

- Create new_user_email_trigger
- Create new_user_sms_trigger
- Create new_user_email_rule and connect to forwardjs-dev/send_email
- Create new_user_sms_rule and connect to forwardjs-dev/send_sms

- Test new_user_email_trigger
- Test new_user_sms_trigger

- Create forwardjs-dev/userchangeseq sequence = upkar-cloudant-package/read → forwardjs-dev/notifyuser
- Create user_changes_trigger
- Create upkar-cloudant-changed-rule and connect to forwardjs-dev/userchangeseq
- Launch Cloudant in IBM Cloud and add a document with “user_email” and “user_phone. You should get email and sms on save.

Web Actions and API

- Convert the identity action into a web action
- Invoke using ic fn
- Get the URL
- Invoke using CURL
- Show in browser
- Send back JSON status
- Send back an image



Follow github exercises for web APIs and build a slack bot using IBMCloud Functions

Web Actions and API

```
function main(params) {  
  return {  
    statusCode: 200,  
    headers: { 'Content-Type': 'application/json' },  
    body: params  
  };  
}
```

- Create action
- Convert to web action
- Invoke the action
- Get the URL
- CURL the URL

Web Actions and API

```
function main() {  
  return {  
    headers: { location: "http://openwhisk.org" },  
    statusCode: 302  
  };  
}
```

```
function main() {  
  let html = "<html><body>Hello World!</body></html>"  
  return { headers: { "Content-Type": "text/html" },  
    statusCode: 200,  
    body: html };  
}
```

```
function main() {  
  let png = "<BASE64 ENCODED IMAGE STRING>"  
  return { headers: { "Content-Type": "image/png" },  
    statusCode: 200,  
    body: png };  
}
```


Web Actions and API

```
~/L/M/c/D/c/o/package_try ic fn api create --help
create a new API
Usage:
  wsk api create ([BASE_PATH] API_PATH API_VERB ACTION] | --config-file CFG_FILE) [flags]

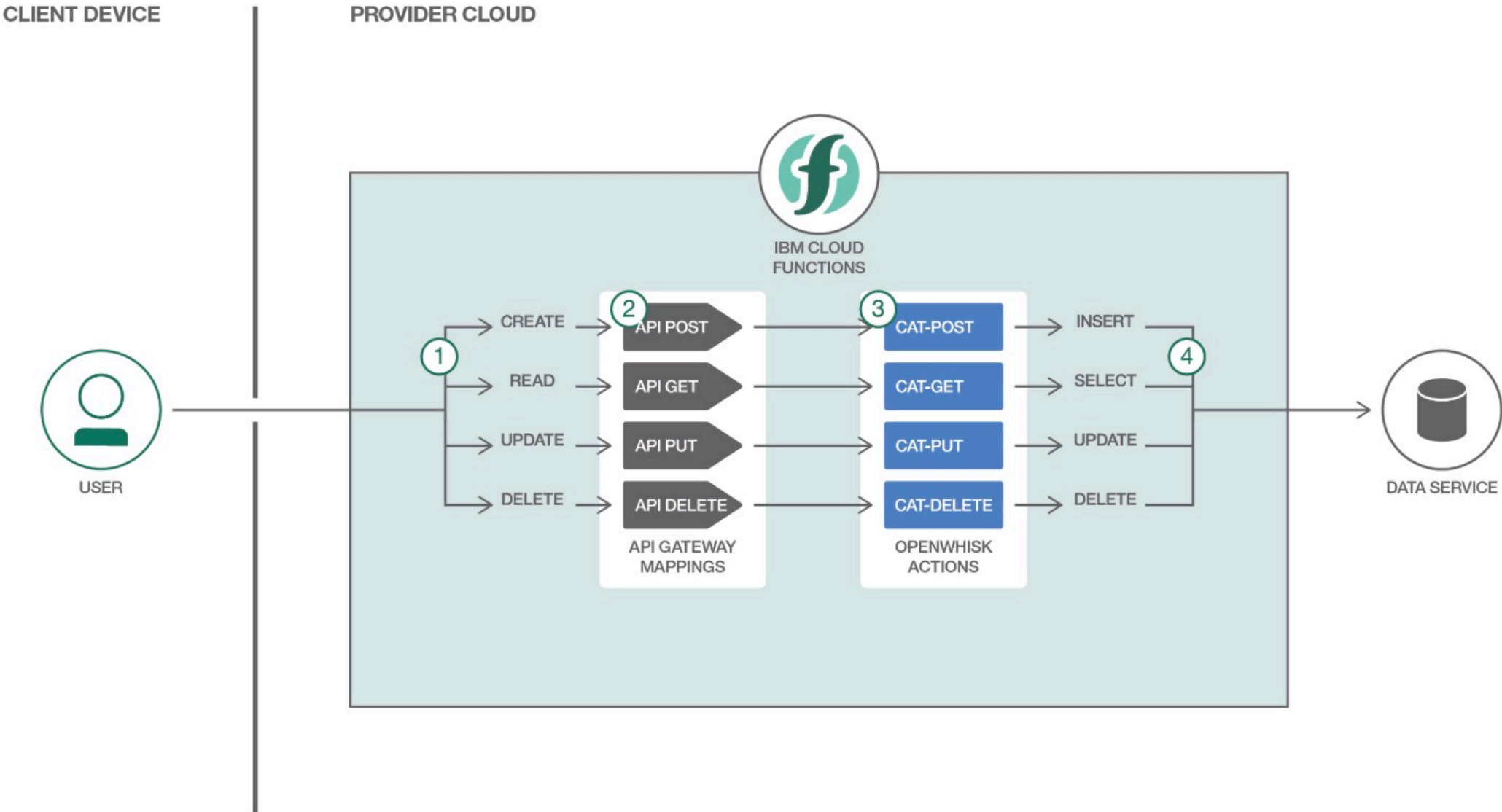
Flags:
  -n, --apiname string      Friendly name of the API; ignored when CFG_FILE is specified (default BASE_PATH)
  -c, --config-file CFG_FILE CFG_FILE containing API configuration in swagger JSON format
  --response-type TYPE      Set the web action response TYPE. Possible values are html, http, json, text, svg (default "json")
```

```
$ ibmcloud wsk api create /api/hello get hello --response-type json --apiname "hello-world"
ok: created API /api/hello GET for action /_/hello
https://service.us.apiconnect.ibmcloud.com/gws/apigateway/api/<UUID>/api/hello
```

```
$ ibmcloud wsk api list
```

```
$ ibmcloud wsk api get / > swagger.json
```

Web Actions and API – REST API



Web Actions and API – REST API

```
return new Promise(function(resolve, reject) {  
  console.log('Connecting to MySQL database');  
  var mysql = require('promise-mysql');  
  var connection;
```

```
if (result[0]) {  
  resolve({  
    statusCode: 200,  
    headers: {  
      'Content-Type': 'application/json'  
    },  
    body: result[0]  
  });  
}
```

```
reject({  
  headers: {  
    'Content-Type': 'application/json'  
  },  
  statusCode: 404,  
  body: {  
    error: "Not found."  
  }  
});
```

```
}).catch(function(error) {  
  if (connection && connection.end) connection.end();  
  console.log(error);  
  reject({  
    headers: {  
      'Content-Type': 'application/json'  
    },  
    statusCode: 500,  
    body: {  
      error: "Error."  
    }  
  });  
});  
);
```

What's Next

💪 Finish the rest of exercises
<https://github.com/lidderupk/openwhisk-workshops>

💪 Participate in the interexchange meetings
<https://www.youtube.coWatm/channel/UCbzgShnQk8F43NKsvEYA1SA>

💪 Clone and follow the numerous IBM Code Patterns
<https://developer.ibm.com/patterns/category/serverless/>

💪 Join Slack
<https://openwhisk.apache.org/slack.html>

💪 Medium
<https://medium.com/openwhisk>

💪 Contribute on Github
<https://github.com/apache?q=openwhisk>

Thank you

Upkar Lidder
Developer Advocate, IBM

- > ulidder@us.ibm.com
- > [@lidderupk](https://twitter.com/lidderupk)
- > blog.upkarlidder.com

IBM Code