Composing IBM Cloud Functions

http://ibm.biz/serverless-research

Kerry Chang, Nick Mitchell,
Rodric Rabbah, Olivier Tardieu

Cloud Functions

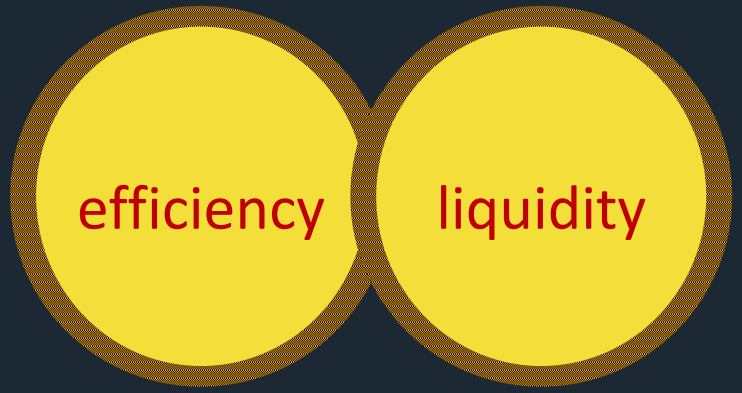
```
// hello.js: JSON dictionary -> JSON dictionary

function main(params) {
  return { message: "Hi " + params.name }
}
```

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

```
$ wsk action invoke hello --param name John --result
{
    "message": "Hi John"
}
```

User’s Perspective

- Benefits
  - auto provisioning: no need to provision or maintain servers
  - auto scaling: fast transparent reactive resource allocation
  - fine-grained billing, pay for load not capacity
  - polyglot: language and runtime agnostic
  - open source, hosted by IBM



- Constraints
  - stateless: no persistence across invocations
  - low limits on code size, payload size, memory footprint, execution time

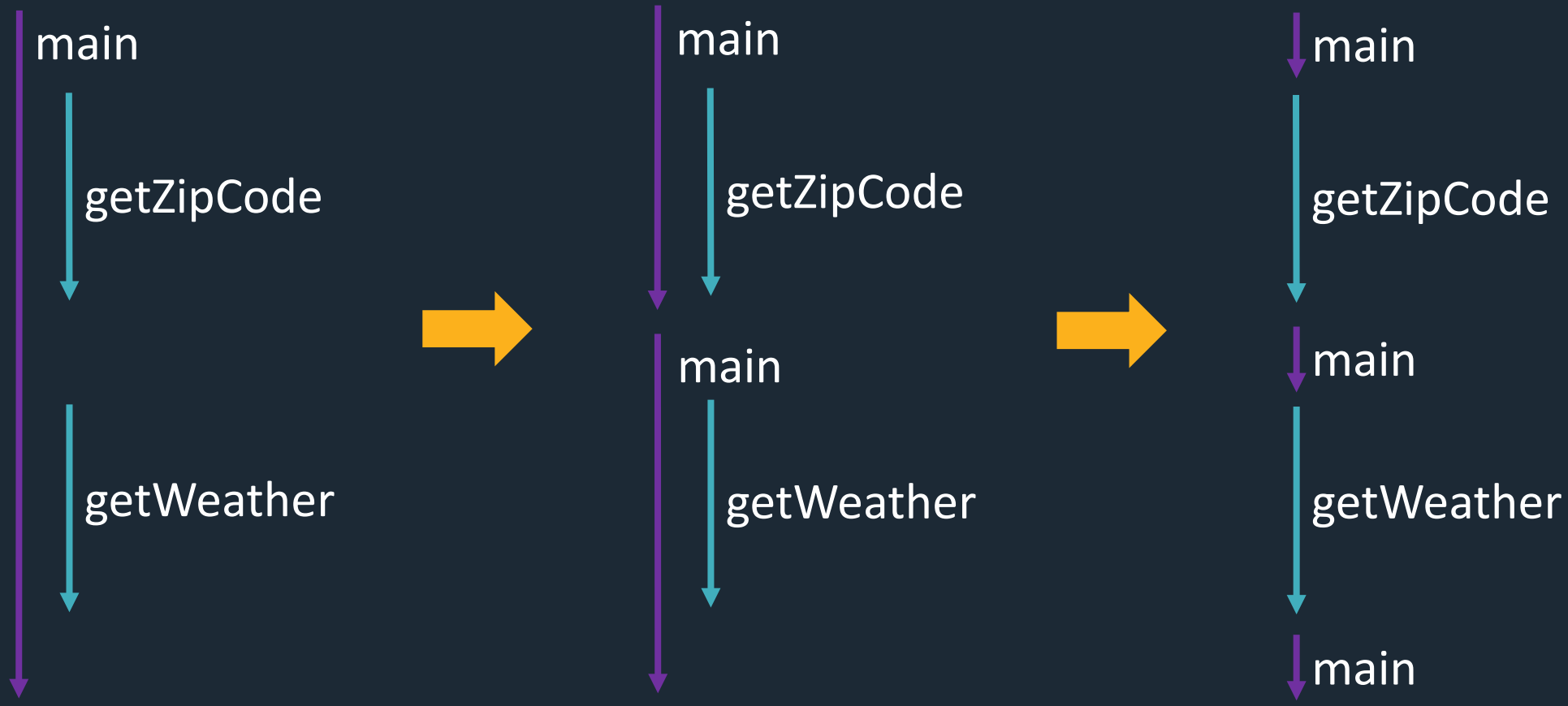
Composition?

- How to build complex applications using simple functions?

- By composing simple functions...

```
try {
    let zipCode = getZipCode(location);
    return getWeather(zipCode);
} catch(err) {
    return { message: `Unable to retrieve weather info: ${err}` };
}
```

Execution Strategy



Continuations

- We need to split to the execution of a composition into a chain of function invocations
- We need the ability to dynamically chain function invocations
  - specify the next function... and the one after that
- We need a compact, polyglot encoding of the execution state of a composition
  - manual: continuation-passing style
  - automatic: checkpoint/restart, compiler/runtime, source-to-source...
  - hybrid

Composition using Composer

```
try {
   let zipCode = getZipCode(location);
   return getWeather(zipCode);
} catch(err) {
   return { message: `Unable to retrieve weather info: ${err}` };
}
```

- same language
- same control flow
- non-capturing lambdas

```
composer.try(
   composer.sequence(
      'getZipCode',          // Cloud Function
      'getWeather'),         // Cloud Function
   (err) => ({ message: `Unable to retrieve weather info: ${err}` })
);
```

Composition Methods

Composition	Example
sequence	composer.sequence('getZipCode', 'getWeather')
conditional	composer.if('authenticate', /* then */ 'welcome', /* else */ 'login')
loop	composer.while('needMoreData', 'fetchMoreData')
error handling	composer.try('DivideByN', /* catch */ 'NaN')
variables (dynamic scoping)	composer.let('n', 42, ...)

Code Generation

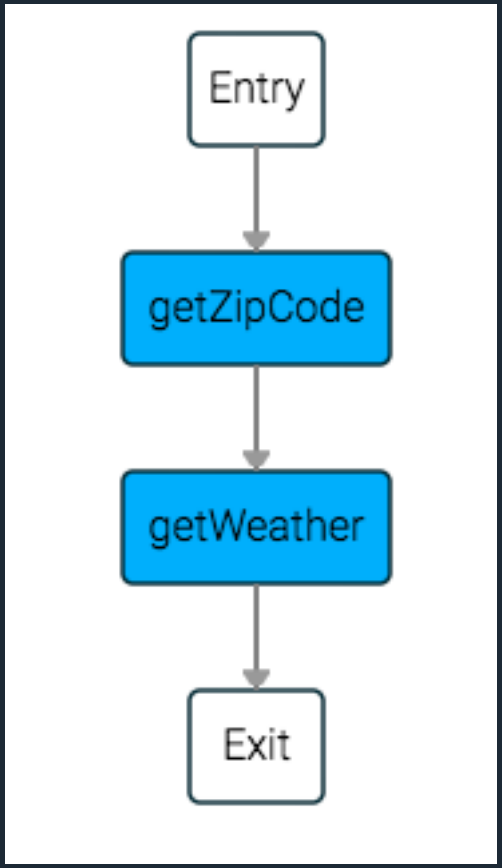
Composition

```
composer.sequence(
  'getZipCode',
  'getWeather');
```

Composer

Automaton

```
{
   "Entry": "action_0",
   "States": {
      "action_0": {
         "Type": "Task",
         "Action": "getZipCode",
         "Next": "action_1"
      },
      "action_1": {
         "Type": "Task",
         "Action": "getWeather"
      }
   },
   "Exit": "action_1"
}
```



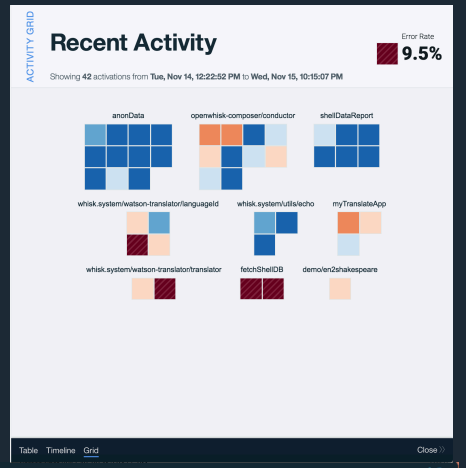
Execution State

- Automaton
- Next control state
- Stack with three kinds of elements
  - anonymous values
  - name, value maps for variables in scope
  - exception handlers in scope

Takeaway

- Composer is a program “language” for the Cloud
  - developer friendly *and* Cloud enabled
  - https://github.com/ibm-functions/composer
- The Cloud is not a typical execution environment
  - limits, pricing...
- The Cloud is an opportunity for PL research!
- IBM Cloud Functions/Apache OpenWhisk is a great research platform

Cloud Function Shell



IBM **Cloud Functions** Shell — OPENWHISK.NG.BLUEMIX.NET VMRC_KERRYSPCHANG HELP

```
ok

> wsk activation get c5f8a4c8d8ac4459b8a4c8d8ac345921
ok

> activation list
```

c5f8a4c8d8ac44…	translator	68ms	failed	Today at 10:15:07 PM
5e836e7c35034e…	entering next task	183ms	ok	10:15:07 PM
b23d73b9748e43…	languageId	578ms	ok	10:15:06 PM
fa3046e93db74c…	echo	3ms	ok	10:15:06 PM
727db52eed5648…	entering next task	163ms	ok	10:15:06 PM
a0d9e1bbccab4d…	languageId	10ms	failed	10:12:37 PM
03df0c2d6d5d47…	myTranslateApp	186ms	ok	10:12:36 PM
c7890babd1d64c…	entering next task	169ms	ok	10:12:36 PM
21dd048aef3042…	conductor	31ms	ok	10:12:22 PM
07a4631f5fe247…	translator	491ms	ok	10:12:21 PM

```
ok

> session list
```

727db52eed5648…	myTranslateApp	1.1s	ok	Today at 10:15:06 PM
c7890babd1d64c…	myTranslateApp	363ms	ok	10:12:36 PM
4c49523e9f3b42…	myTranslateApp	1.6s	ok	10:12:20 PM
3269b6094ce846…	myTranslateApp	3.3s	ok	10:11:56 PM

```
ok

> enter your command
```

IBM **Cloud Functions** Shell — OPENWHISK.NG.BLUEMIX.NET VMRC_KERRYSPCHANG HELP

c5f8a4c8d8ac4459…	translator	68ms	failed
5e836e7c35034ebb…	entering next task	183ms	ok
b23d73b9748e4333…	languageId	578ms	ok
fa3046e93db74ce9…	echo	3ms	ok
727db52eed564821…	entering next task	163ms	ok
a0d9e1bbccab4d73…	languageId	10ms	failed
03df0c2d6d5d4708…	myTranslateApp	186ms	ok
c7890babd1d64c44…	entering next task	169ms	ok
21dd048aef304271…	conductor	31ms	ok
07a4631f5fe2475d…	translator	491ms	ok

```
ok

> session list
```

727db52eed564821bdb…	myTranslateApp	1.1s	ok
c7890babd1d64c44890…	myTranslateApp	363ms	ok
4c49523e9f3b42ce895…	myTranslateApp	1.6s	ok
3269b6094ce8465ba9b…	myTranslateApp	3.3s	ok

```
ok

> session get 4c49523e9f3b42ce89523e9f3bb2ce30
ok

> session get 4c49523e9f3b42ce89523e9f3bb2ce30
ok

> enter your command
```

SESSION

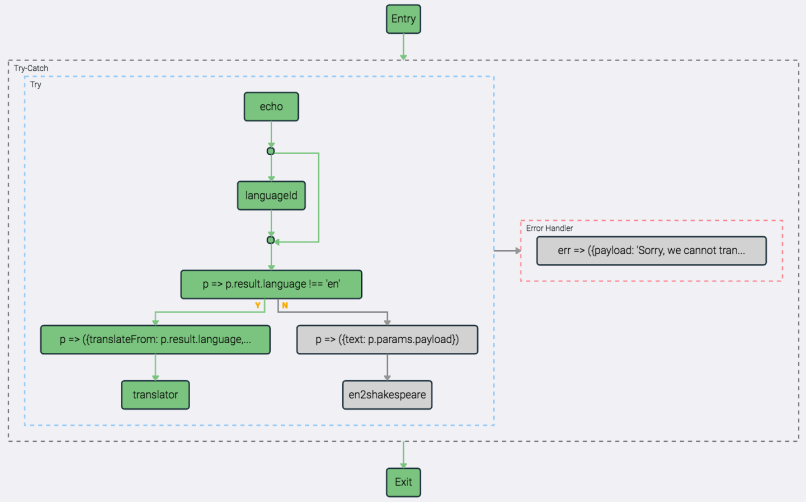
4c49523e9f3b42ce89523e9f3bb2ce30

myTranslateApp

This activation started **Today at 10:12:20 PM**

1.6s

~$6.80 per million



Result Trace Raw Session Flow — Close