Third International Workshop on Serverless Computing (WoSC) 2018

In conjunction with IEEE CLOUD 2018 affiliated with 2018 IEEE World Congress on Services (IEEE SERVICES 2018)

Agenda Overview

8:30	**to**	**10:00**	**a.m.**	**Welcome and Keynote**
10:00	**to**	**10:30**	**a.m.**	**Break**
10:30	**to**	**12:00**	**p.m.**	**Papers**
12:00	**to**	**1:30**	**p.m.**	**Lunch break**
1:30	**to**	**3:00**	**p.m.**	**Invited speakers**
3:00	**to**	**3:30**	**p.m.**	**Break**
3:30	**to**	**5:00**	**p.m.**	**Invited speaker and panel**

Latest agenda, abstracts, speakers, and links to slides posted here:

https://www.serverlesscomputing.org/wosc3/#program

Your Feedback

Please tell us what works and what does not work.

Google feedback form:

https://goo.gl/forms/vTDGOvuKED4FJJA22

Your questions during feedback: Docs and Slack

Workshop program does not leave much time for questions. Please post your questions during workshop to Google doc and join #wosc3 slack channel to discuss: https://docs.google.com/document/d/1AjGEx7sLoVX7FxjpRL3X3QjzYg0bBlNgXyXqaMWx8os/edit?usp=sharing

We will have discussion during panel at the end of workshop and panelists will be checking the slack channel and google do. Sign up for slack channel in browser open **wosc-tutorial-invite.mybluemix.net**

https://wosc-tutorial-invite.mybluemix.net/

Then open slack server https://future-compute.slack.com and join #wosc3

Workshop purpose:

"Many of the major cloud vendors, have released serverless platforms within the last two years, including Amazon Lambda, Google Cloud Functions, Microsoft Azure Functions, IBM Cloud Functions. There is, however, little attention from the research community.

This workshop brings together researchers and practitioners to discuss their experiences and thoughts on future directions.”

Organization

Latest:
https://www.serverlesscomputing.org/wosc3

Previous workshops

Second International Workshop on Serverless Computing (WoSC) 2017 in Las Vegas, NV, USA on December 12th, 2017 part of Middleware 2017.

First International Workshop on Serverless Computing (WoSC) 2017 in Atlanta, GA, USA on June 5th, 2017 part of ICDCS 2017.

Presentations from previous workshops:

https://www.serverlesscomputing.org

Workshop co-chairs

Paul Castro, IBM Research

Vatche Ishakian, Bentley University

Vinod Muthusamy, IBM Research

Aleksander Slominski, IBM Research

Steering Committee

Geoffrey Fox, Indiana University

Dennis Gannon, Indiana University & Formerly Microsoft Research

Arno Jacobsen, MSRG (Middleware Systems Research Group)

Program Committee

Gul Agha, University of Illinois at Urbana-Champaign

Azer Bestavros, Boston University

Flavio Esposito, Saint Louis University

Rodrigo Fonseca, Brown University

Ian Foster, University of Chicago and Argonne National Laboratory

Geoffrey Fox, Indiana University

Dennis Gannon, Indiana University & Formerly Microsoft Research

Arno Jacobsen, MSRG (Middleware Systems Research Group)

Tyler Harter, GSL, Microsoft

Pietro Michiardi, Eurecom

Peter Pietzuch, Imperial College

Rodric Rabbah, IBM Research

Rich Wolski, University of California, Santa Barbara

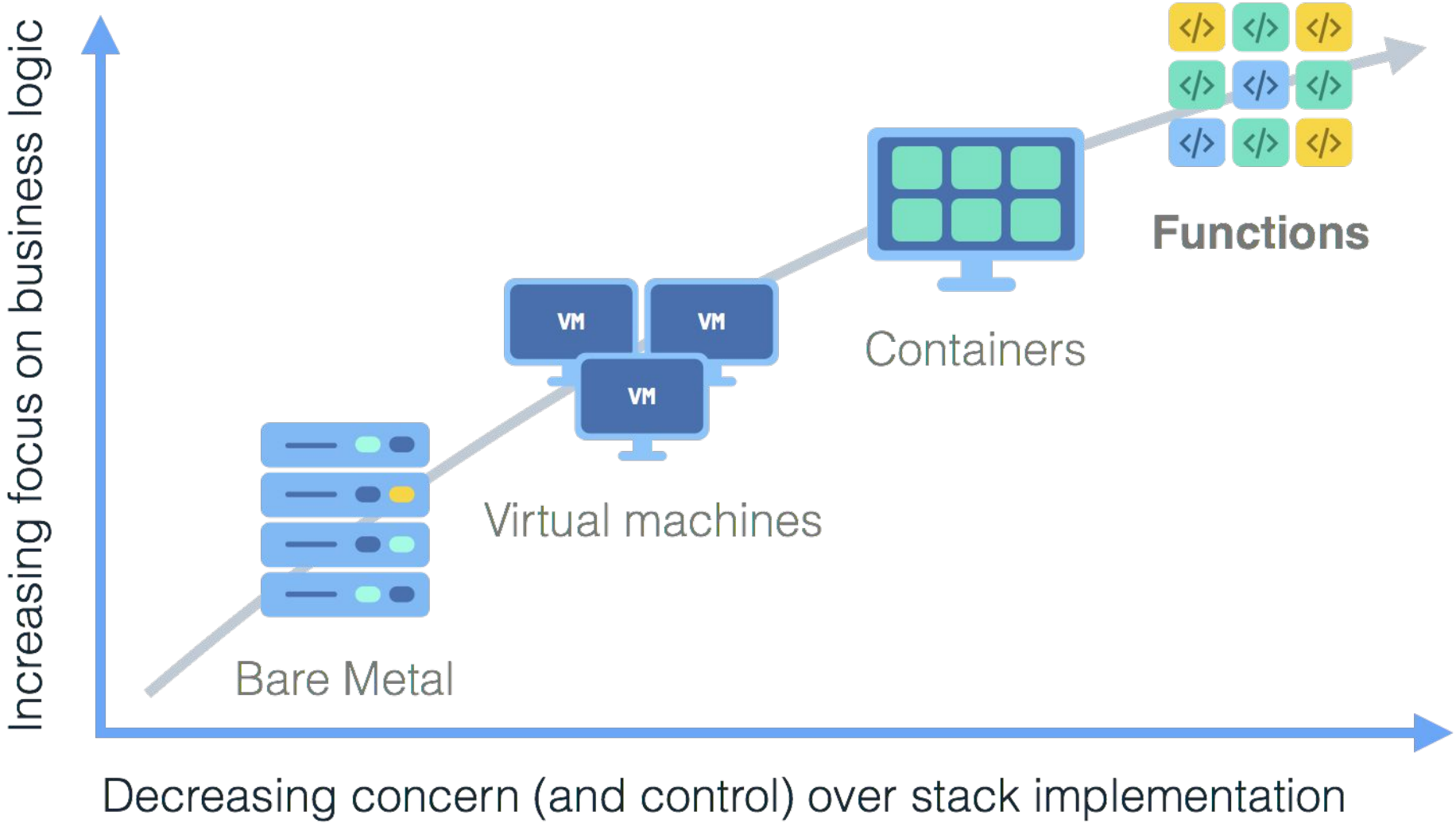
Fourth International Workshop on Serverless Computing (WoSC)

Part of 11th IEEE/ACM International Conference on Utility and Cloud Computing (UCC) and 5th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)

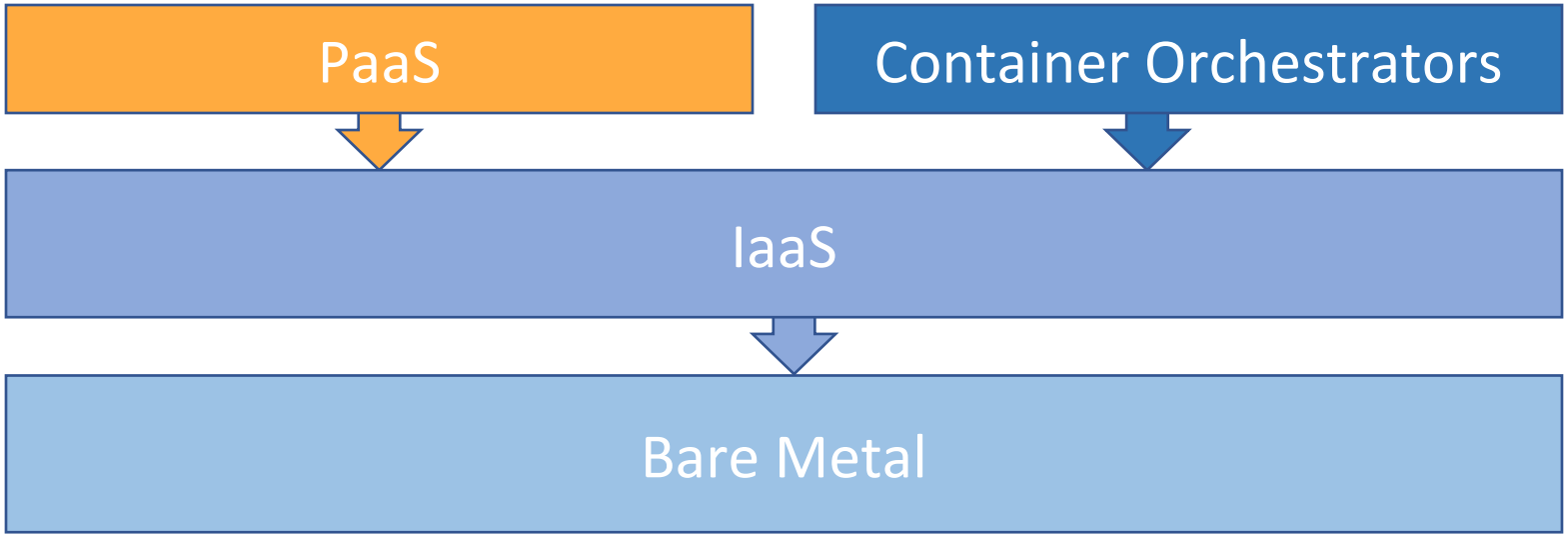
Paper Submission: September 01, 2018

Conference: December 17-20, 2018 in Zurich, Switzerland

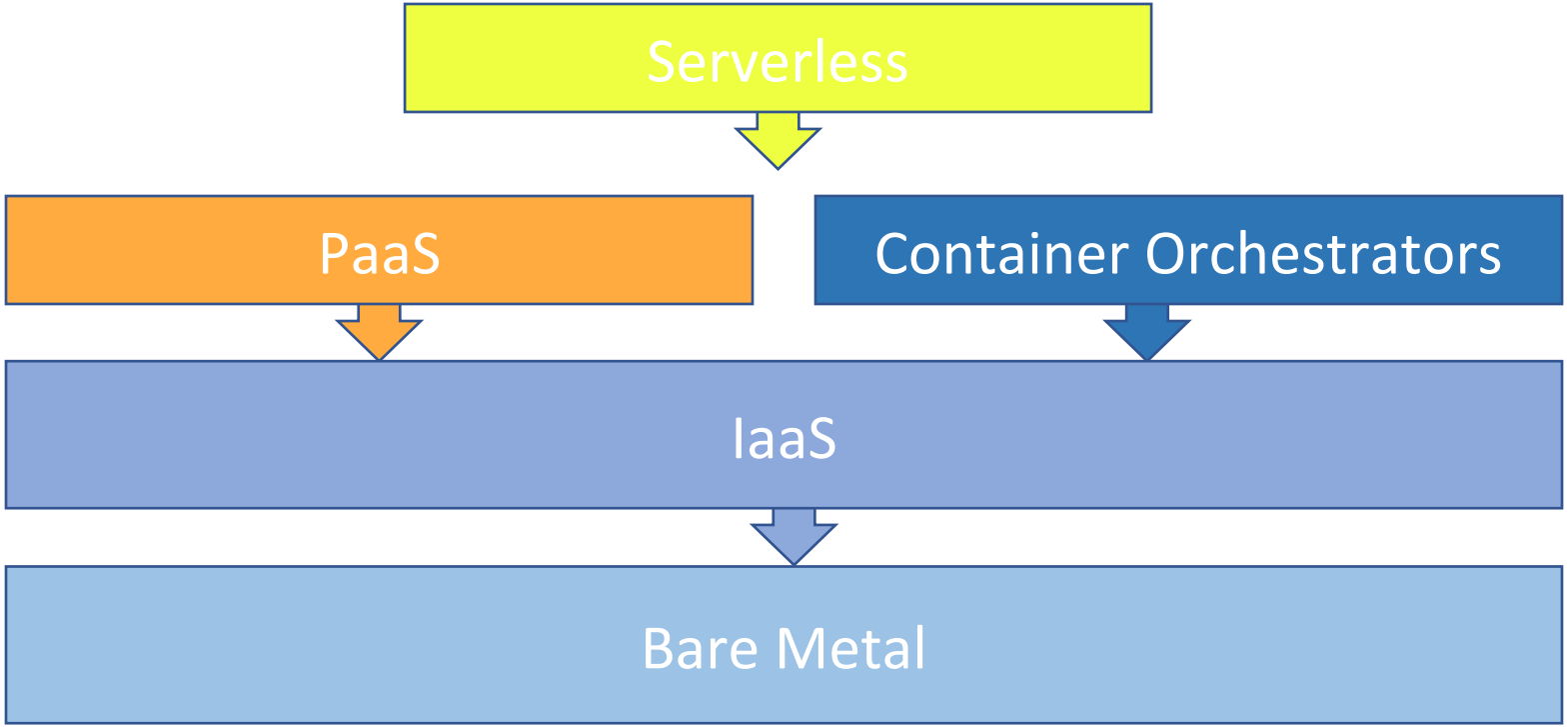
Serverless in 5 minutes

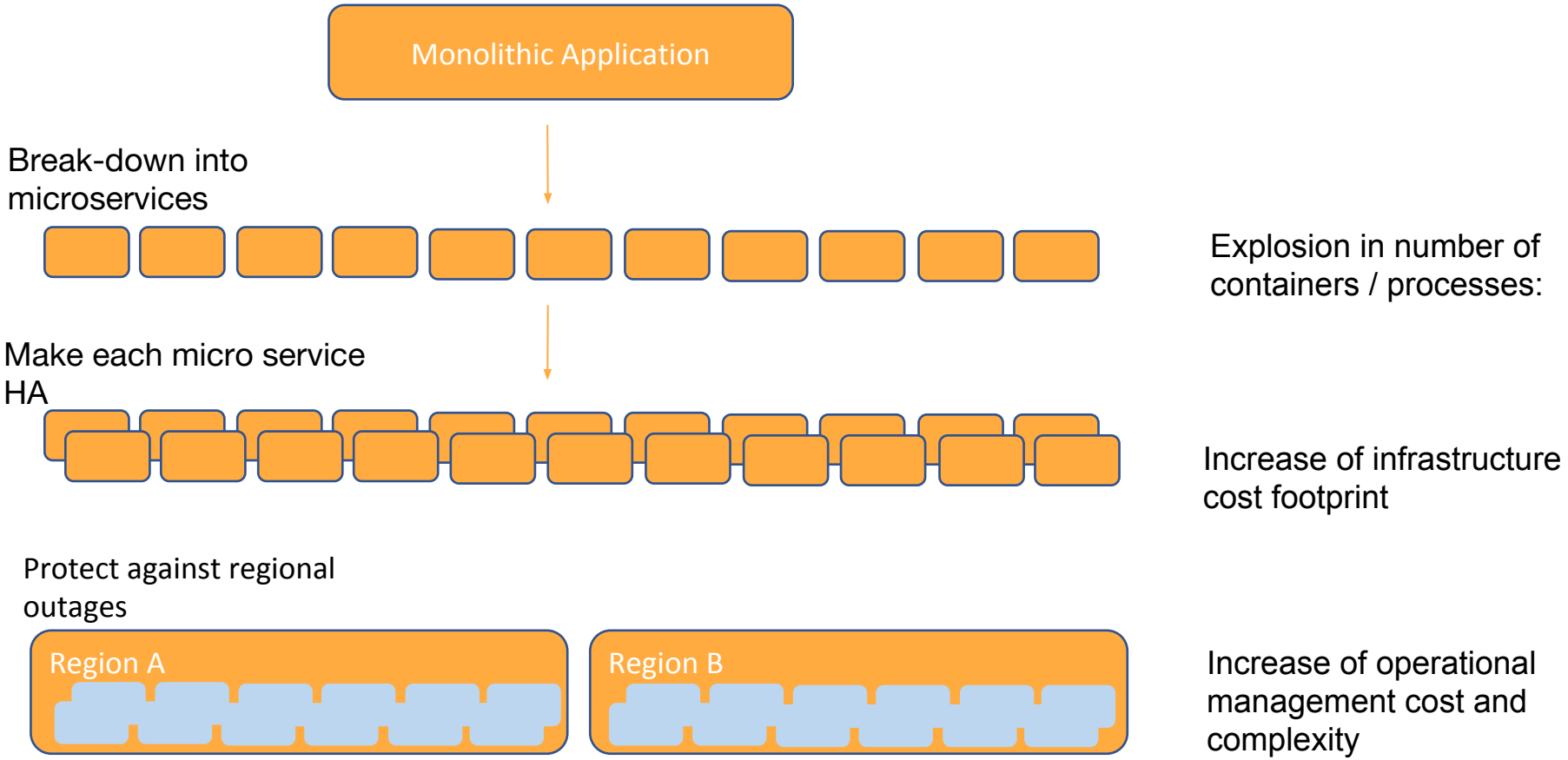

Increasing focus on business logic
Bare Metal
VM
VM
VM
Virtual machines
Containers
Functions
Decreasing concern (and control) over stack implementation

Evolution Of Serverless



Enter Serverless




Monolithic Application
Break-down into microservices
Explosion in number of containers / processes:
Make each micro service HA
Increase of infrastructure cost footprint
Protect against regional outages
Region A
Region B
Increase of operational management cost and complexity

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

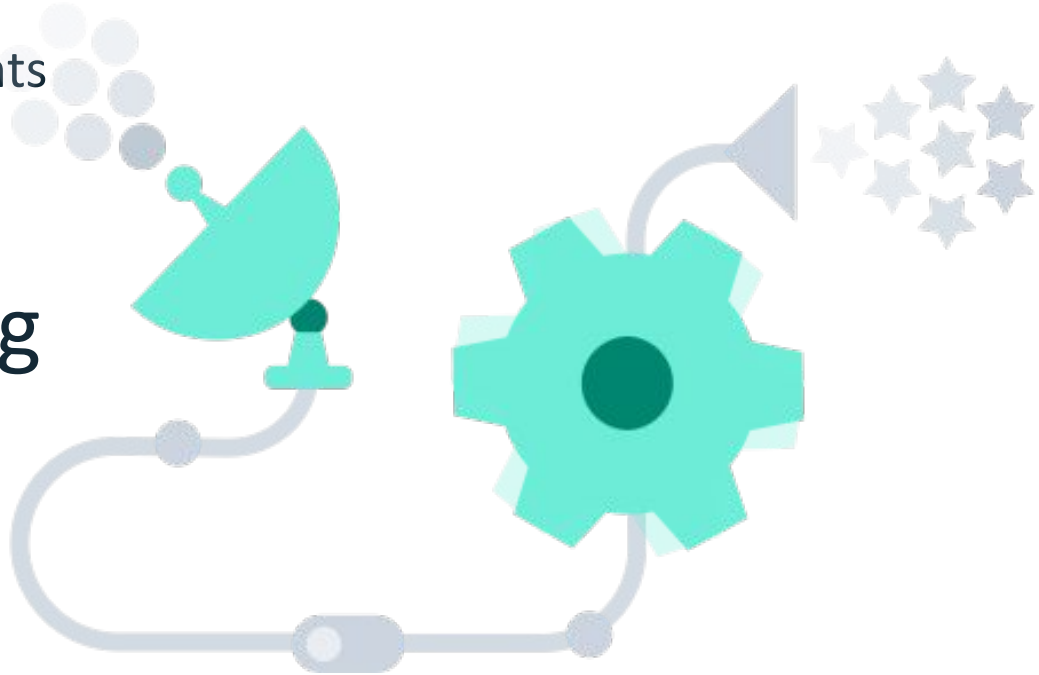
No servers

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

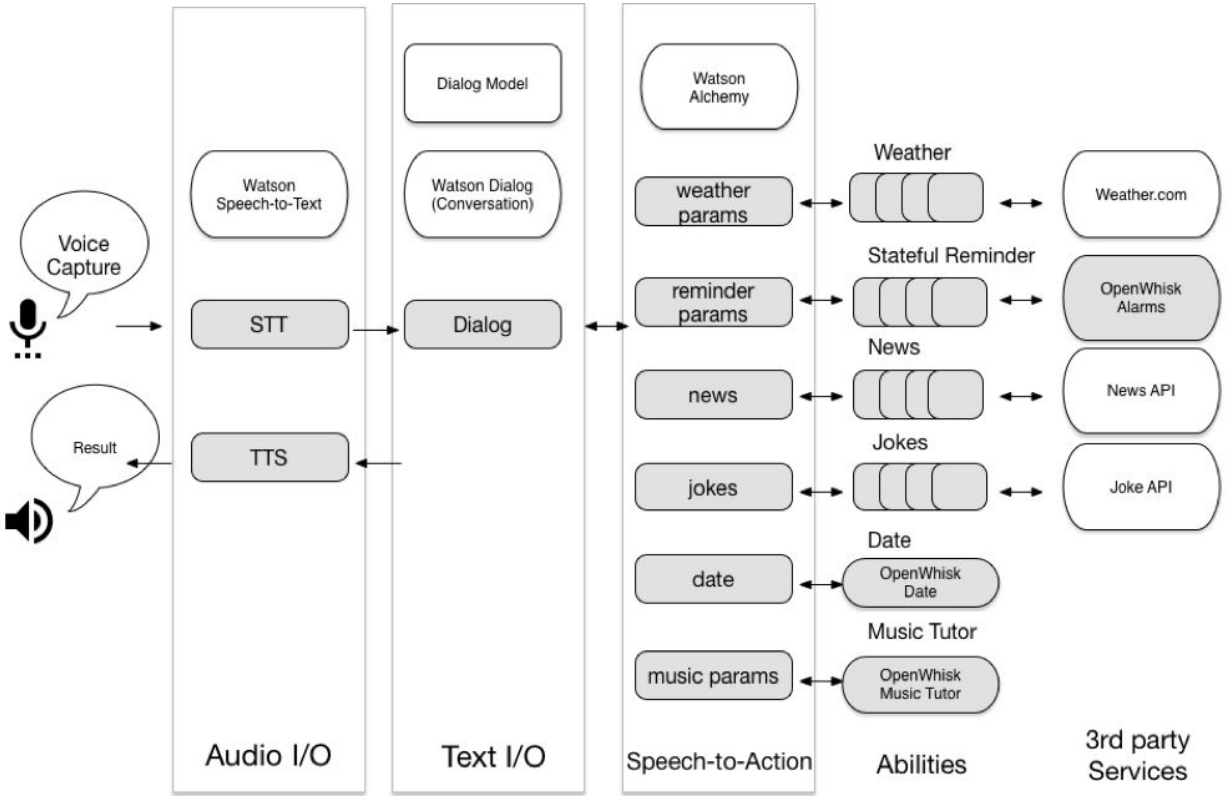
	On-prem	VMs	Containers	Serverless
Time to provision	Weeks-months	Minutes	Seconds-Minutes	Milliseconds
Utilization	Low	High	Higher	Highest
Charging granularity	CapEx	Hours	Minutes	Blocks of milliseconds

Source: Jason McGee, IBM; Serverless Conference 2017.

Key factors for infrastructure cost savings

	Traditional models (CF, containers, VMs)	Serverless
High Availability	At least 2-3 instances of everything	No incremental infrastructure
Multi-region deployment	One deployment per region	No incremental infrastructure
Cover delta between short (<10s) load spikes and valleys (vs average)	~2x of average load	No incremental infrastructure
Example incremental costs	2 instances x 2 regions x 2 = 8x	1x

Chatbots



Building a Chatbot with Serverless Computing, Yan, Mengting and Castro, Paul and Cheng, Perry and Ishakian, Vatche, Proceedings of the 1st International Workshop on Mashups of Things and APIs 2016

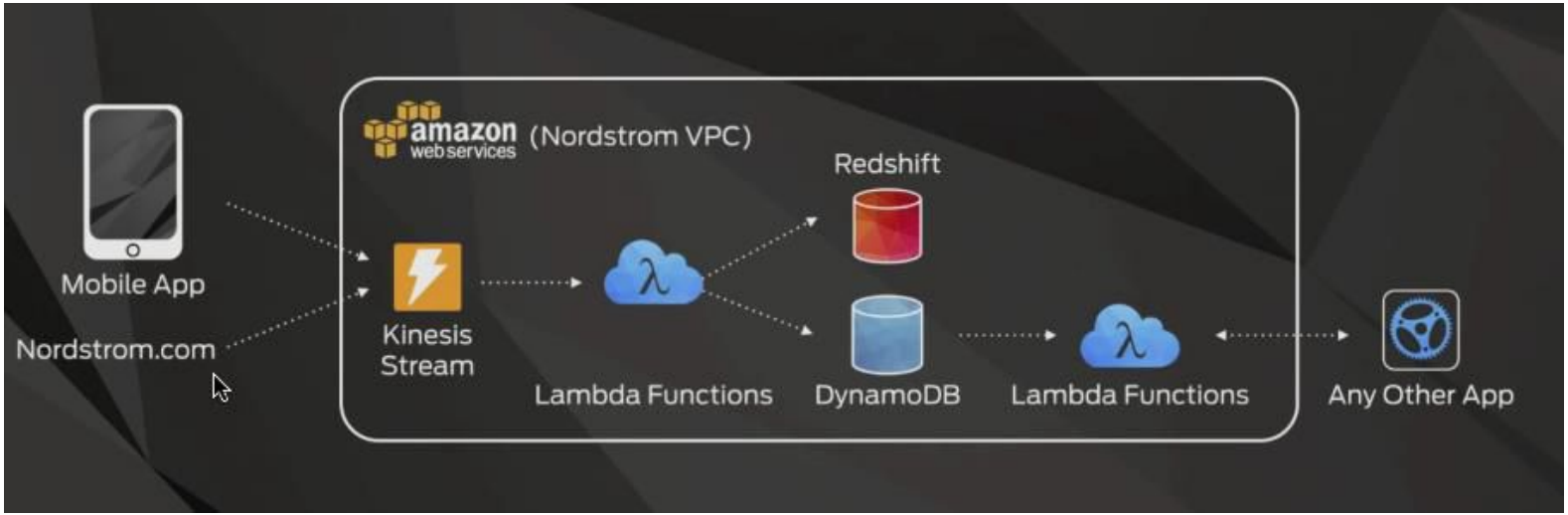
PyWren: a massive data framework for Lambda

- Open Source MapReduce framework using Lambda
- Word count job on 83M items is only 17% slower than PySpark running on dedicated servers.
- Sort 1TB data in 3.4 minutes (Spark 100TB in 23 min)

https://github.com/pywren/pywren
http://pywren.io/

Occupy the Cloud: Distributed Computing for the 99%, Eric Jonas, Qifan Pu, Shivaram Venkataraman, Ion Stoica, Benjamin Recht, https://arxiv.org/abs/1702.04024

Nordstrom Recommendations



15-20 minutes of processing → now in seconds

2x order of magnitude for cost savings

What is Serverless good for?

Serverless is **good** for

short-running
stateless
event-driven

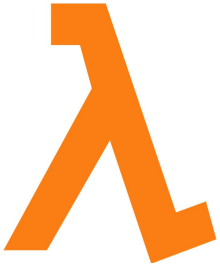
- Microservices
- Mobile Backends
- Bots, ML Inferencing
- IoT
- Modest Stream Processing
- Service integration

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



AWS
Lambda

OpenLambda

Azure Functions

Kubernetes

IBM Cloud Functions

Red-Hat

Google Functions

Tutorial Part 0

Account setup

Slack channel

Sign up for slack channel: **wosc-tutorial-invite.mybluemix.net**

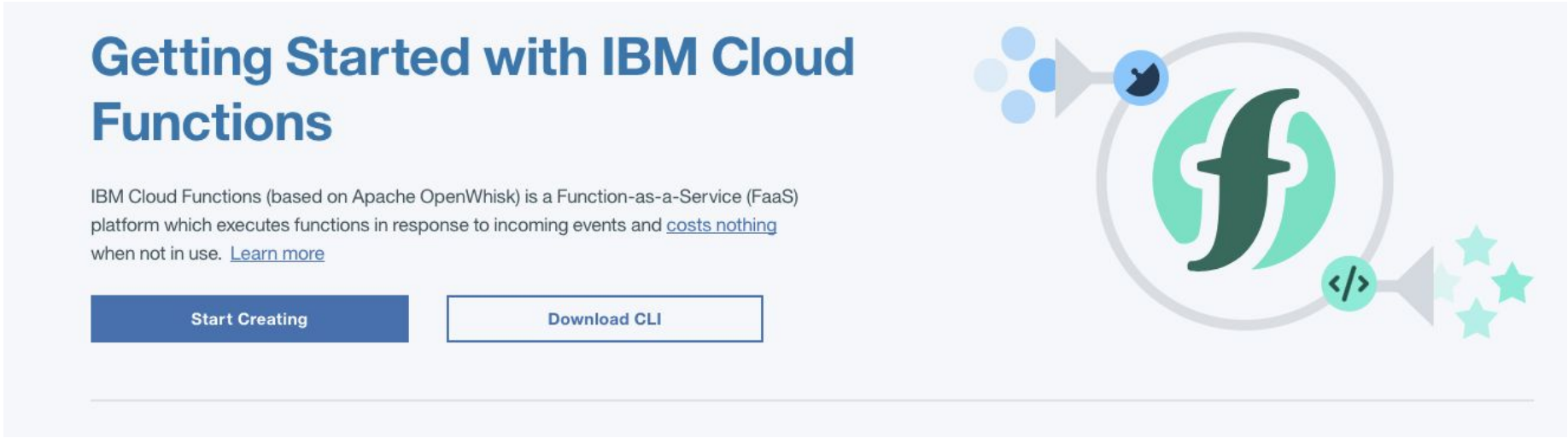
https://wosc-tutorial-invite.mybluemix.net/

Tutorial Setup

slides are available at https://goo.gl/QpD6fi

- Log-in with your IBM Cloud (Bluemix) account: create one if you do not yet have one by clicking the sign-up link or by directly navigating to https://developer.ibm.com/dwblog/2017/building-with-ibm-watson and select “IBM Cloud Lite” to get IBM Cloud (Bluemix) account
- To use OpenWhisk proceed as follows: open a browser window and navigate to https://console.ng.bluemix.net/openwhisk/
- CLick “Start Creating” to create cloud functions directly from browser
- Click the Download OpenWhisk command line tools for your operating system: https://console.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

1. Download

Download and install the Bluemix CLI.

2. Install the Cloud Functions Plugin

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

Copy

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
```

Copy

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
```

Copy

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

For more detail, consult the online Cloud Functions documentation.

Using bx command line

```
bx login -a api.ng.bluemix.net
```

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

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

Using wsk command line (Apache OpenWhisk)

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 middleware --blocking --result

output:

{ "message": "middleware" }

Part 1

OpenWhisk actions, triggers

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

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

Creating and invoking JavaScript actions

- List the actions you created

  ```
  bx wsk action list
  ```

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

  ```
  bx wsk action invoke --blocking hello
  ```

- You can retrieve the list of activations at any time

  ```
  bx wsk activation list
  ```

- Enter the invocation ID shown, for example:

  ```
  bx wsk activation get dde9212e686f413bb90f22e79e12df74
  ```

- You can delete an action

  ```
  bx 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

```
bx wsk action create hello2 hellowithparams.js
```

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

```
wsk action invoke -b hello2 -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

```
bx wsk action create yahooWeather weather.js
```

```
bx wsk action invoke --blocking --result
yahooWeather --param location "Brooklyn, NY"
```

bx wsk action invoke --blocking --result yahooWeather --param location "Las Vegas"

```
{
    "msg": "It is 42 degrees in Las Vegas and Sunny"
}
```

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
    - Sign up for slack channel: **wosc-tutorial-invite.mybluemix.net**

https://wosc-tutorial-invite.mybluemix.net/

- 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