

SI Project presentation

Use OpenWhisk to create a
function-as-a-service (FaaS)
environment



Auchabie Antoine
Ballu Timothée
Paul Yves-Marie
Pescio Vincent
Rouleau Valentin
Trémenbert Alban



Instructions

Goal: Use OpenWhisk to create a function-as-a-service (FaaS) environment

Requirements:

- Code written in Java, Python, Node.js, and PHP must be supported by the FaaS cloud
- External resources such as RabbitMQ and MariaDB must supported



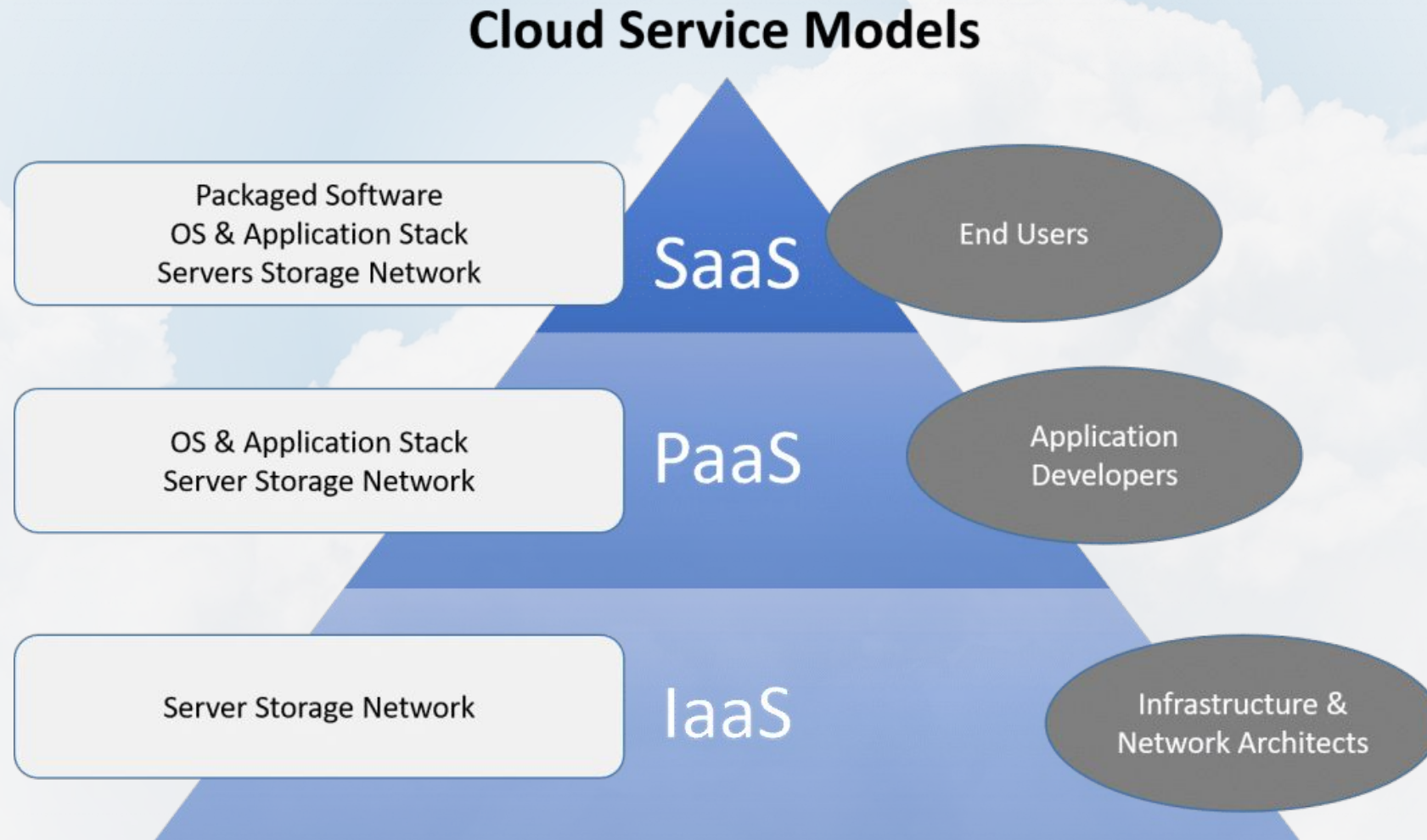
OpenWhisk

Let's talk about ...

The background

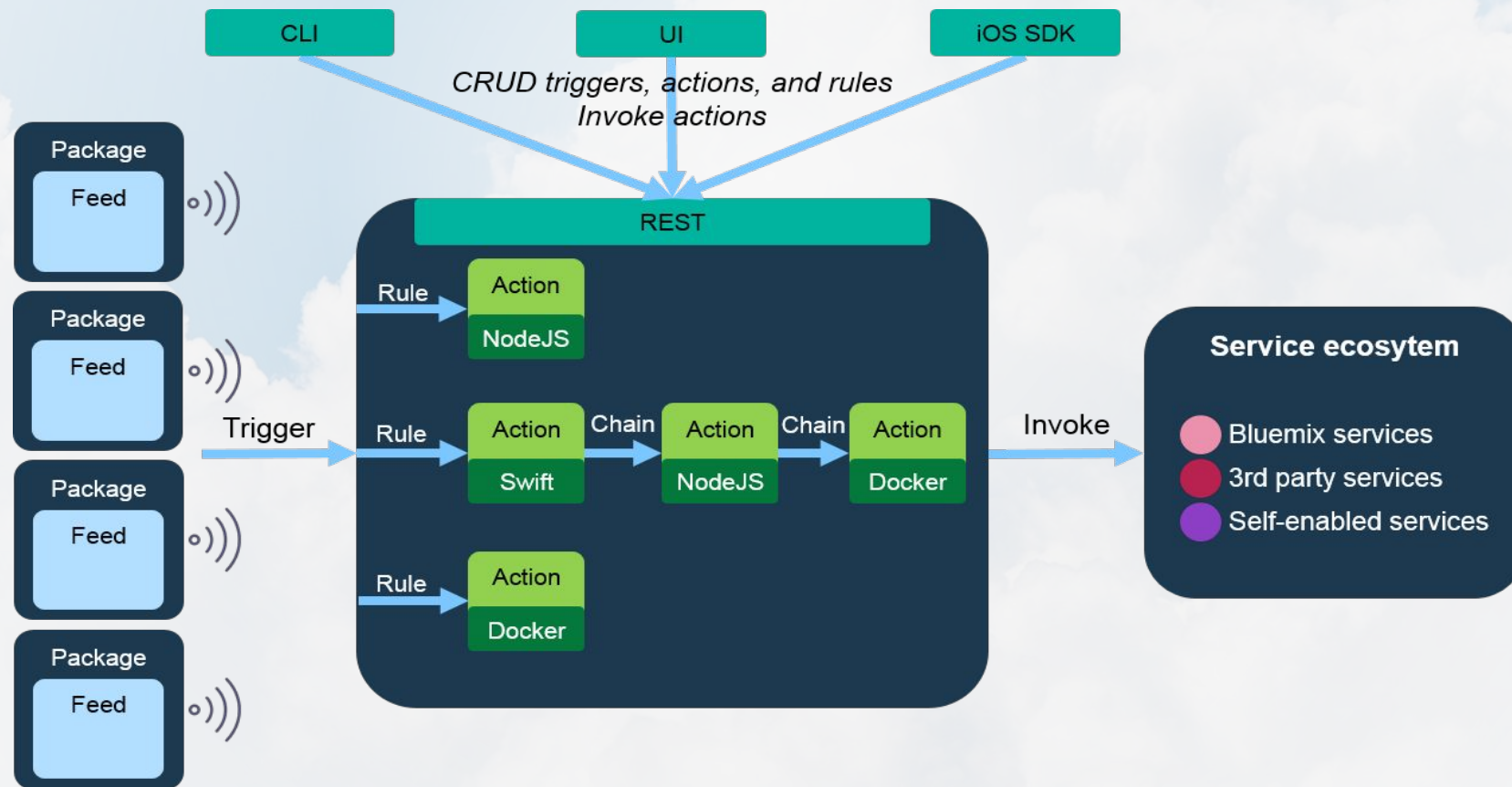
1

Cloud computing model

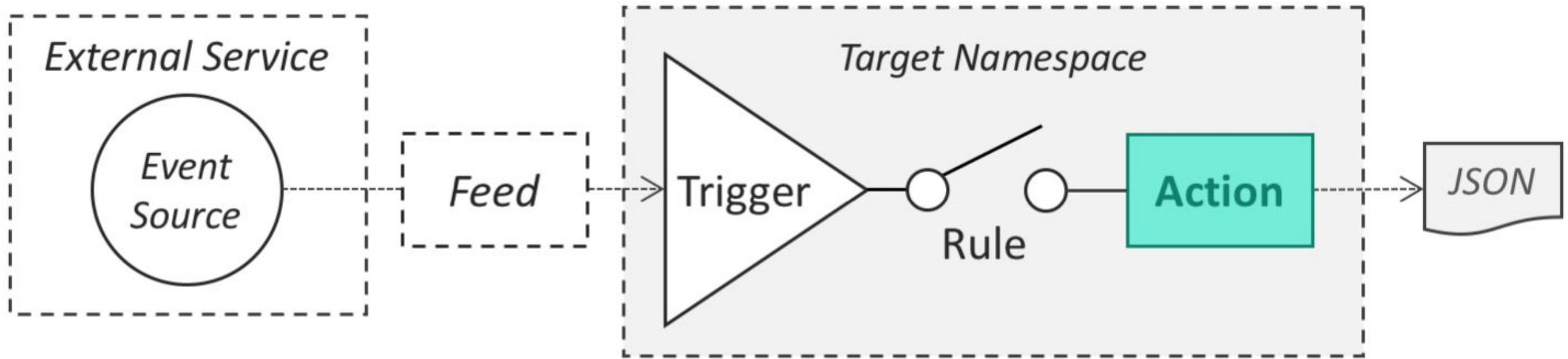


OpenWhisk architecture and definition

- Apache OpenWhisk is an open source serverless cloud platform that executes functions in response to events at any scale.



OpenWhisk programming model



Schema of OpenWhisk programming model



Let's talk about ...

Specification and design

2

Specification and design

- ▶ We created our Faas environment locally on our machines.
- ▶ For the local development, we used Kubernetes which is a container orchestration platform that automates the deployment, scaling, and management of containerized applications.

Specification and design

- ▶ First step : Create a Kubernetes cluster to support an OpenWhisk deployment.
- ▶ Second step : For local development with Kubernetes, we used Helm charts to deployed OpenWhisk.
- ▶ Third step : We decided to create a simple function in Java, Python, Node.js and PHP. These functions will take one or two parameters and will return a simple json object.

Let's talk about ...

Implementation

3

Implementation

- Creation of a Kubernetes cluster (minikube) and initialization of Helm

```
Invite de commandes
Microsoft Windows [version 10.0.17134.706]
(c) 2018 Microsoft Corporation. Tous droits réservés.

C:\Users\roule>minikube start --memory 4096 --cpus 2 --kubernetes-version v1.10.5
minikube v1.0.1 on windows (amd64)
Downloading Kubernetes v1.10.5 images in the background ...
Creating virtualbox VM (CPUs=2, Memory=4096MB, Disk=20000MB) ...
"minikube" IP address is 192.168.99.100
Configuring Docker as the container runtime ...
Version of container runtime is 18.06.3-ce
Waiting for image downloads to complete ...
Preparing Kubernetes environment ...
Pulling images required by Kubernetes v1.10.5 ...
Unable to pull images, which may be OK: pull command is not supported by kubeadm v1.10.5
Launching Kubernetes v1.10.5 using kubeadm ...
Waiting for pods: apiserver proxy etcd scheduler controller dns
Configuring cluster permissions ...
Verifying component health .....
kubectrl is now configured to use "minikube"
Done! Thank you for using minikube!

C:\Users\roule>minikube ssh -- sudo ip link set docker0 promisc on

C:\Users\roule>
```

```
Invite de commandes

C:\Users\roule>helm init
$HELM_HOME has been configured at C:\Users\roule\.helm.

Tiller (the Helm server-side component) has been installed into your Kubernetes Cluster.

Please note: by default, Tiller is deployed with an insecure 'allow unauthenticated users' policy.
To prevent this, run 'helm init' with the --tiller-tls-verify flag.
For more information on securing your installation see: https://docs.helm.sh/using_helm/#securing-your-helm-installation
Happy Helming!

C:\Users\roule>kubectrl create clusterrolebinding tiller-cluster-admin --clusterrole=cluster-admin --serviceaccount=kube-system:default
clusterrolebinding.rbac.authorization.k8s.io "tiller-cluster-admin" created

C:\Users\roule>
```

Implementation

- Deployment of OpenWhisk using the mycluster.yml file

```
CA Invite de commandes
C:\Users\roule\incubator-openwhisk-deploy-kube\helm\openwhisk>helm install . --namespace=openwhisk --name=isep -f mycluster.yaml
NAME: isep
LAST DEPLOYED: Sat May 11 19:14:26 2019
NAMESPACE: openwhisk
STATUS: DEPLOYED

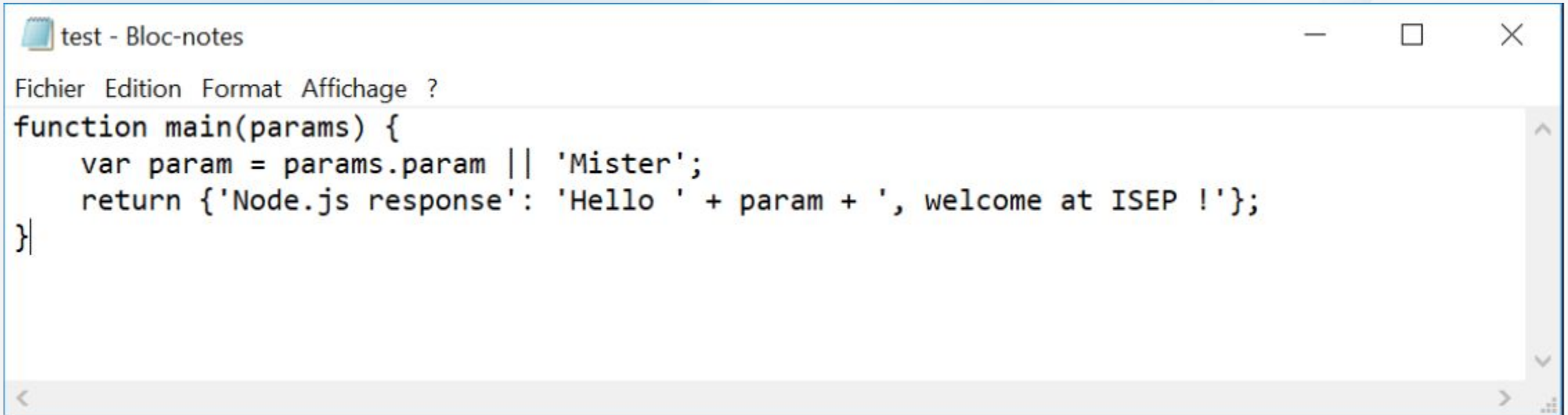
RESOURCES:
==> v1/ConfigMap
NAME                                DATA  AGE
isep-db.config                      10     6s
isep-init-couchdb                   1      6s
isep-install-packages-cm            1      6s
isep-invoker-scripts                 2      6s
isep-nginx                           1      6s
isep-tests-package-checker           1      6s
isep-tests-smoketest                 1      6s
isep-whisk.config                    9      6s
isep-zookeeper                       1      6s

==> v1/DaemonSet
NAME            DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE SELECTOR  AGE
isep-invoker    1        1        0      1            0           <none>         5s

==> v1/Deployment
```

Implementation

- Functions to test our Faas environnement



```
test - Bloc-notes
Fichier Edition Format Affichage ?
function main(params) {
  var param = params.param || 'Mister';
  return {'Node.js response': 'Hello ' + param + ', welcome at ISEP !'};
}
```

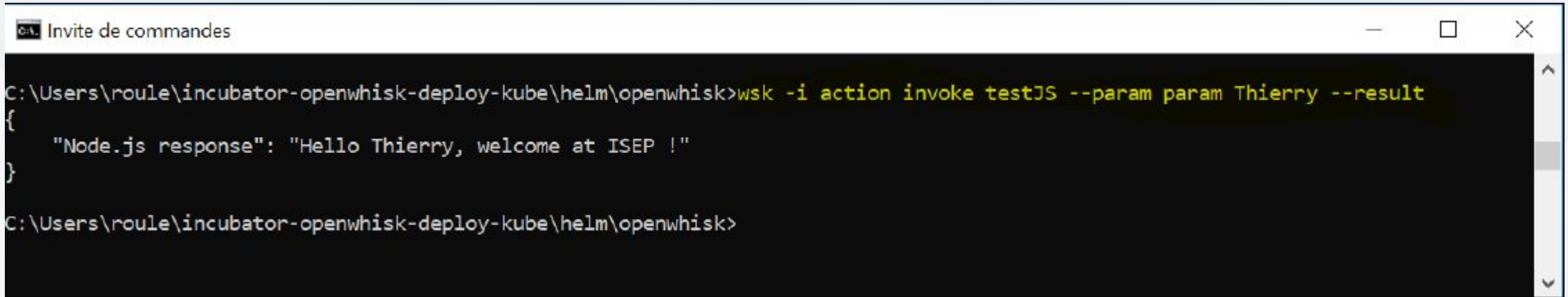



Let's talk about ...

Results and evaluation

4

Results and evaluation



```
C:\Users\roule\incubator-openwhisk-deploy-kube\helm\openwhisk>wsk -i action invoke testJS --param param Thierry --result
{
  "Node.js response": "Hello Thierry, welcome at ISEP !"
}

C:\Users\roule\incubator-openwhisk-deploy-kube\helm\openwhisk>
```

- ▶ Our actions return the expected results.
- ▶ We can assure that our OpenWhisk FaaS environment is correctly deployed and works
- ▶ We also can affirm that code written in Java, Node.js, Python and PHP is supported by our FaaS environment

Conclusion

THANK YOU !

Questions ?