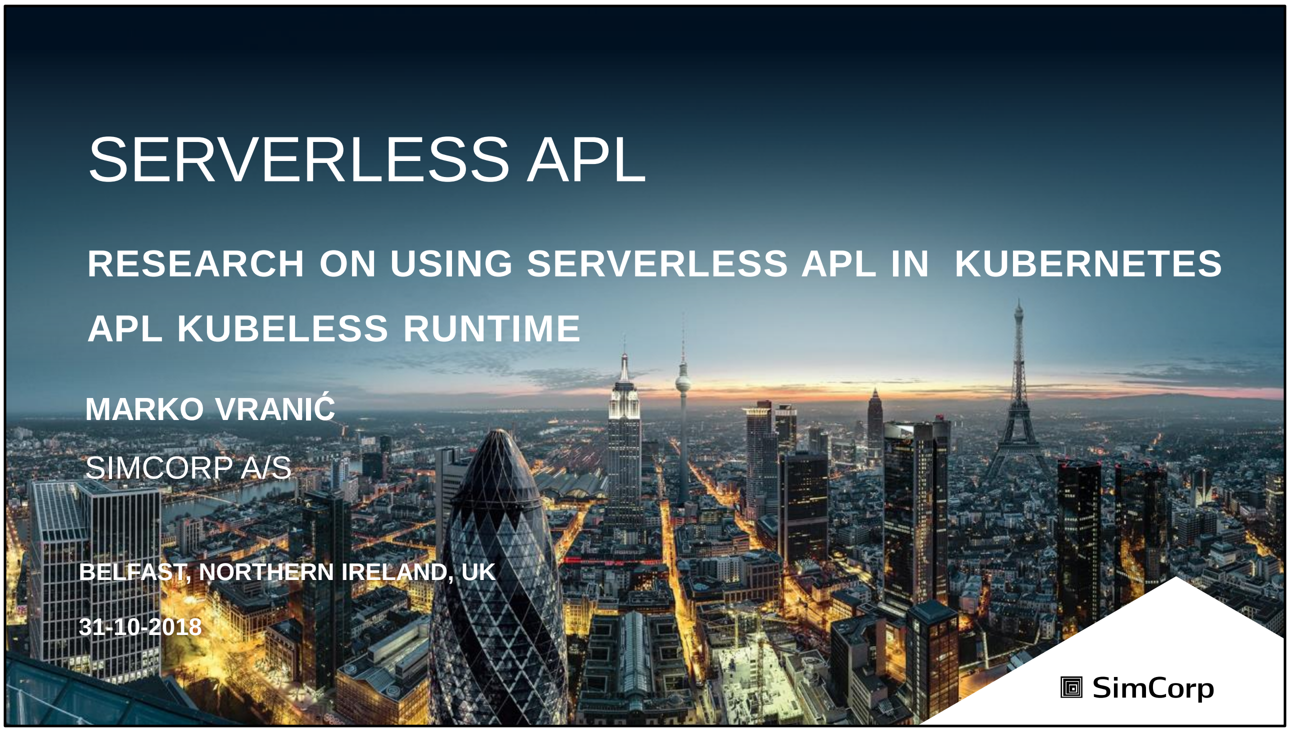


SERVERLESS APL

RESEARCH ON USING SERVERLESS APL IN KUBERNETES

APL KUBELESS RUNTIME

MARKO VRANIĆ

SIMCORP A/S

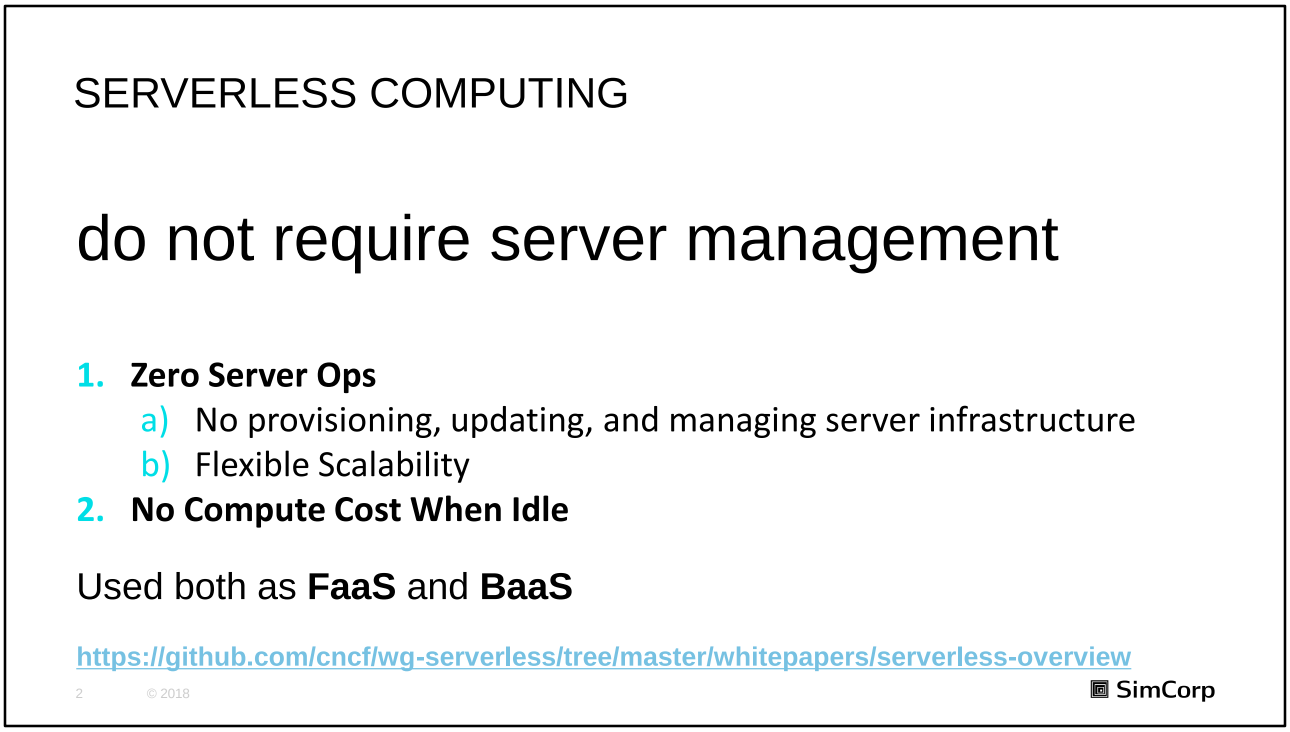
BELFAST, NORTHERN IRELAND, UK

31-10-2018

SimCorp

For now this is just research in Cloud technologies in SimCorp A/S.

SimCorp is the world's leading provider of integrated investment management solutions. There is around 180+ clients around the world.


SERVERLESS COMPUTING

do not require server management

1. **Zero Server Ops**
   a) No provisioning, updating, and managing server infrastructure
   b) Flexible Scalability
2. **No Compute Cost When Idle**

Used both as **FaaS** and **BaaS**

https://github.com/cncf/wg-serverless/tree/master/whitepapers/serverless-overview

2 © 2018 SimCorp

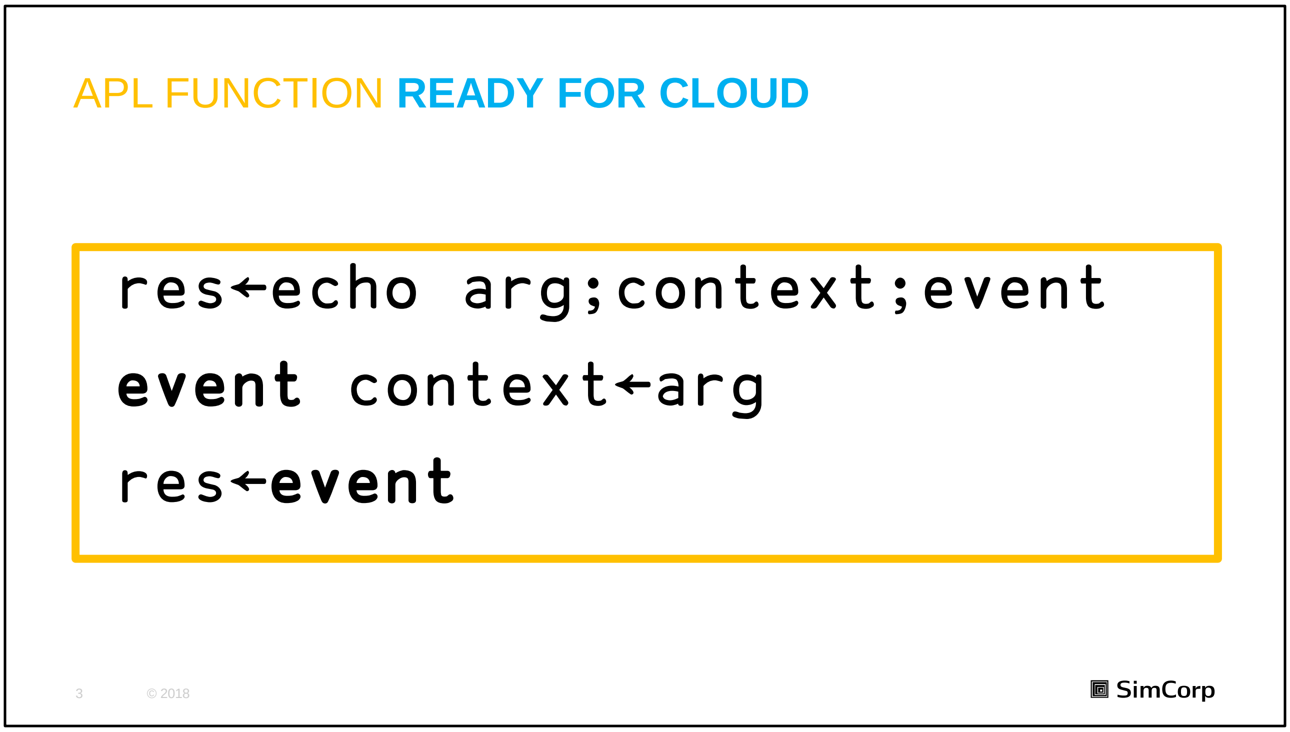
Two serverless personas:

1. **Developer:** writes **code** for, and benefits from the serverless platform which provides them the point of view that there are no servers nor that their code is always running.
2. **Provider**: **deploys** the serverless platform for an external or internal customer

Serverless provides:

1. **Functions-as-a-Service (FaaS),** which typically provides event-driven computing.
2. **Backend-as-a-Service (BaaS),** which are third-party API-based services that replace core subsets of functionality in an application

https://github.com/cncf/wg-serverless/blob/master/workflow/spec/spec.md



APL FUNCTION **READY FOR CLOUD**

```
res←echo arg;context;event
event context←arg
res←event
```

3 © 2018

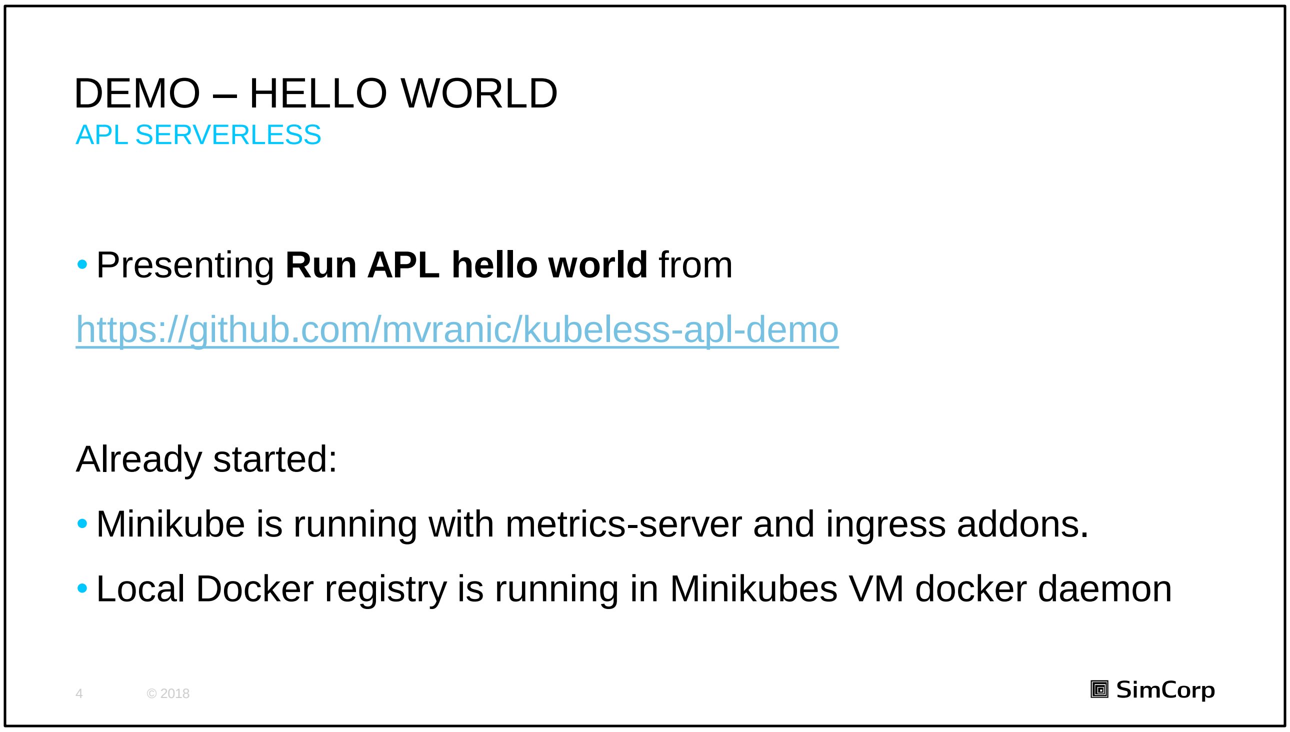
SimCorp

Basic APL serverless function.

Developer has focus on code i.e. function.

Set scene. **Difference from Morten Kromberg is just the code which is just deployed to cloud.** The developer has focus only on the code.

Extend with `⎕TS in example.`



DEMO – HELLO WORLD

APL SERVERLESS

- Presenting **Run APL hello world** from

https://github.com/mvranic/kubeless-apl-demo

Already started:

- Minikube is running with metrics-server and ingress addons.
- Local Docker registry is running in Minikubes VM docker daemon

4 © 2018 SimCorp

Demo is available on GitHub on my account mvranic.

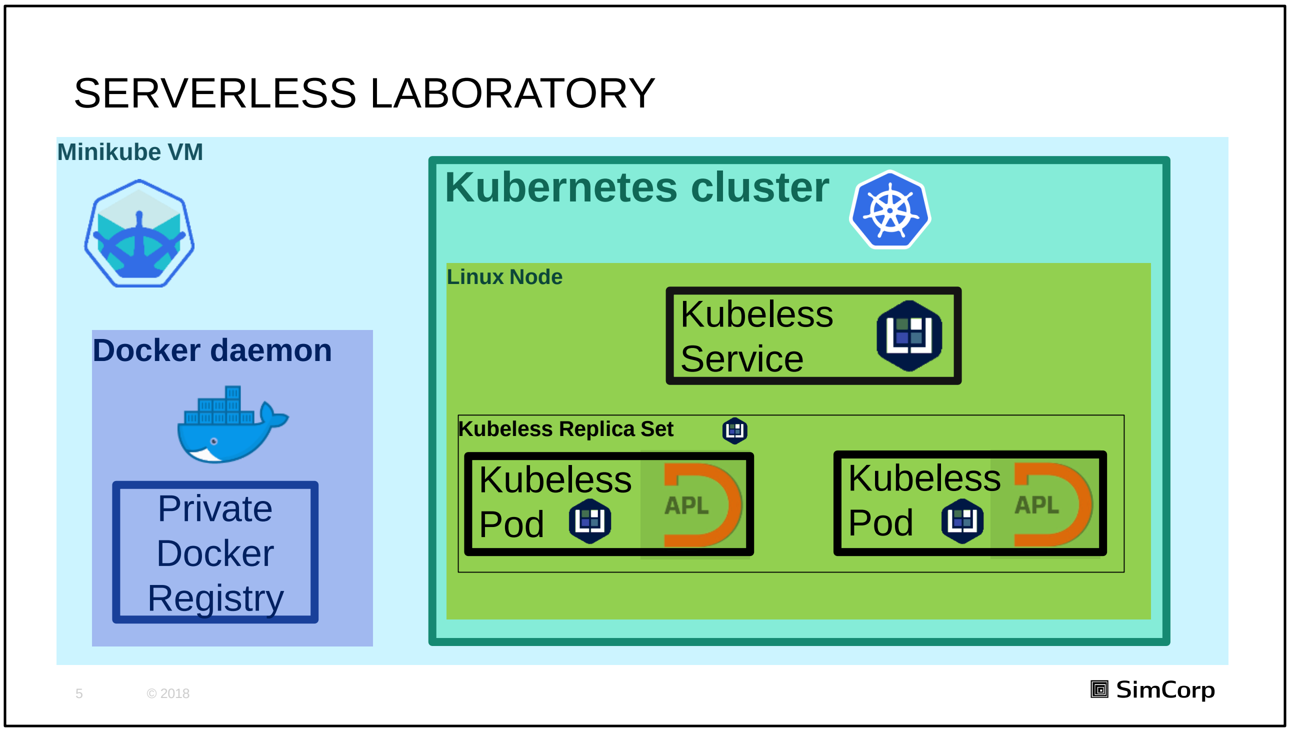
Click on https://github.com/mvranic/kubeless-apl-demo

You can try it.

Steps:

1. Got power shell which is at: C:\gitrepos\mvranic\kubeless-apl-demo\src
2. code .

   ➔This start VS Code



SERVERLESS LABORATORY

5 © 2018 SimCorp

Experiment:

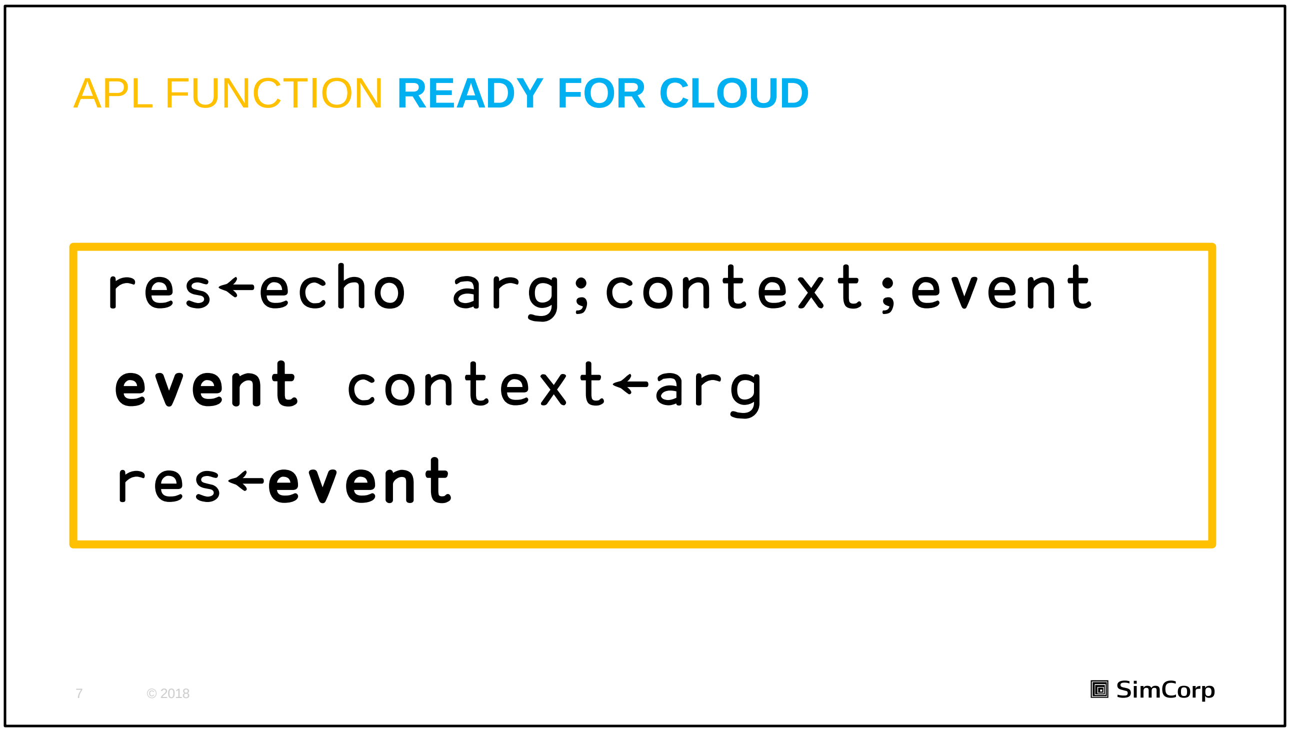
To run APL functions in Kubeless serverless framework, where Kubernetes cluster is run locally in Minikube, where a local Docker registry is installed

Why Kubeless? It is extension of k8s and I needed 3 minutes to run simple example.

Now I will show all servers where serverless is ruining.

EXPERIMENT

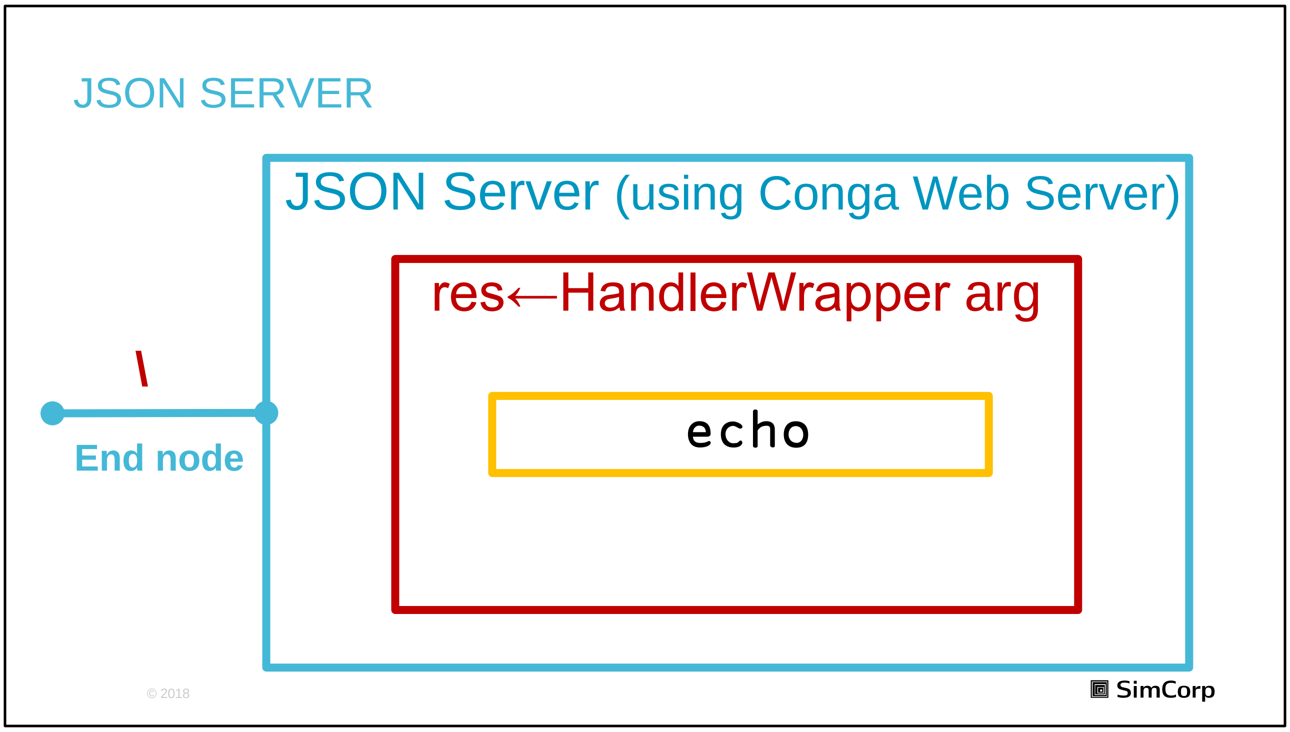
To run APL functions in Kubeless serverless framework, where Kubernetes cluster is run locally in Minikube, where a local Docker registry is installed


APL FUNCTION **READY FOR CLOUD**

```
res←echo arg;context;event
event context←arg
res←event
```

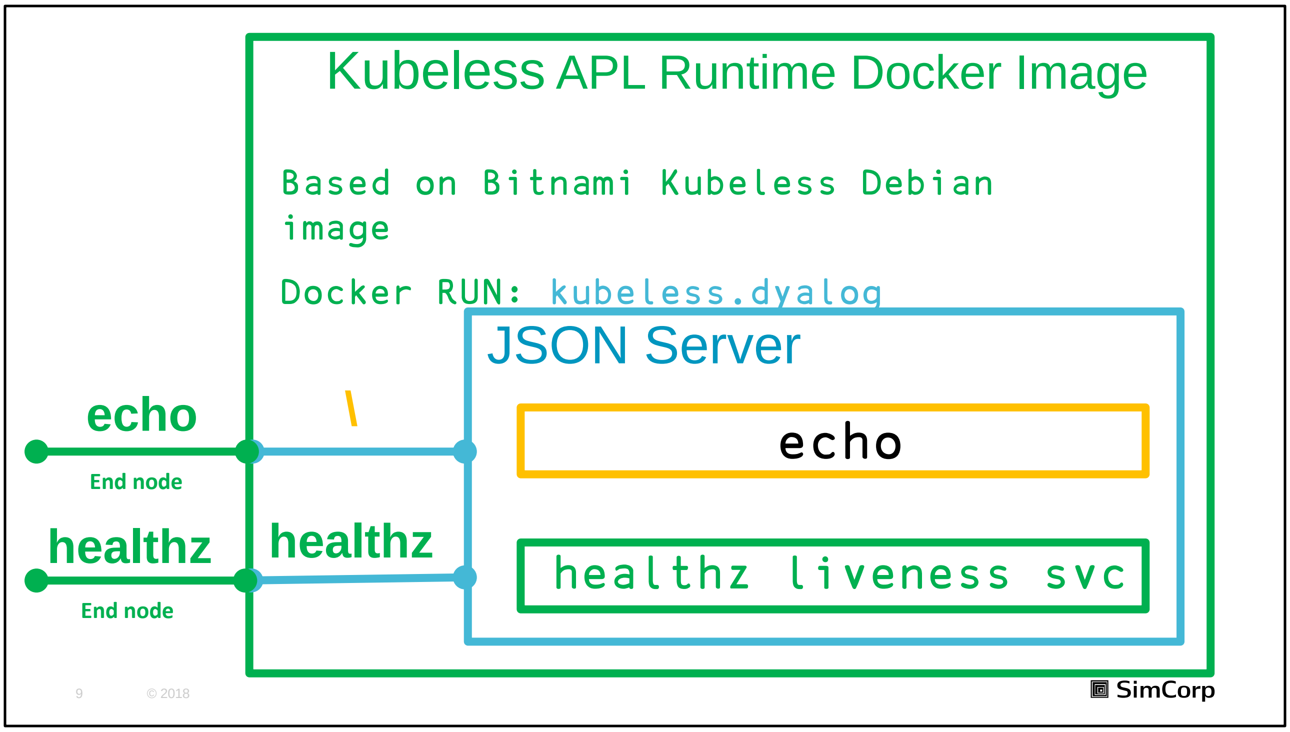
SimCorp

Which can run Dyalog APL interpreter.



There are echo node

Simple REST servers which servers JSON enabled services.


Kubeless APL Runtime Docker Image
Based on Bitnami Kubeless Debian image
Docker RUN: kubeless.dyalog
JSON Server
echo
\
echo
End node
healthz
healthz
healthz liveness svc
End node
9
© 2018
SimCorp

Kubeless Runtime is Docker image.

Image is based standard Bitnami Kubeless Debian image.

Where is installed Dyalog APL 17.0

Where is cloned JSON Server

JSON server is configured

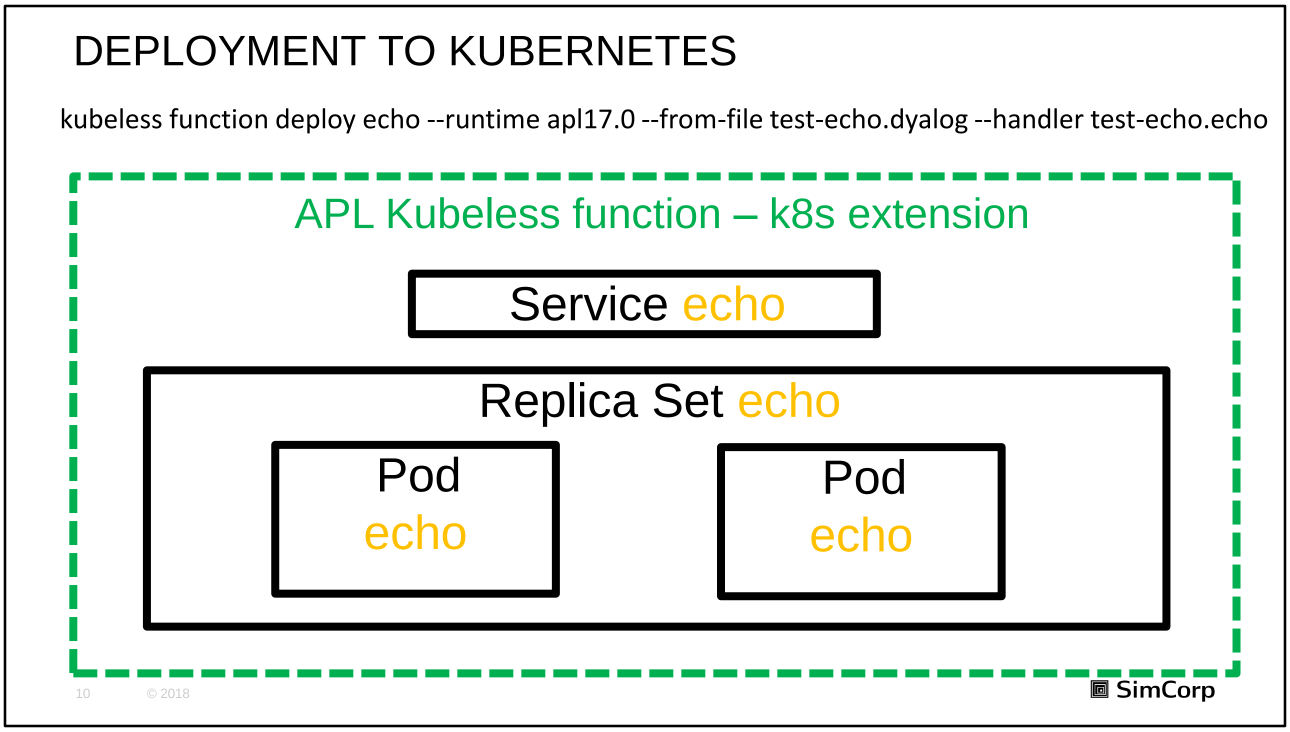
Startup of runtime is implemented

➔Coping of APL code

➔Start JSON Server

Important is Liveliness or health check probe is added to JSONServer

Handler /



DEPLOYMENT TO KUBERNETES

```
kubeless function deploy echo --runtime apl17.0 --from-file test-echo.dyalog --handler test-echo.echo
```

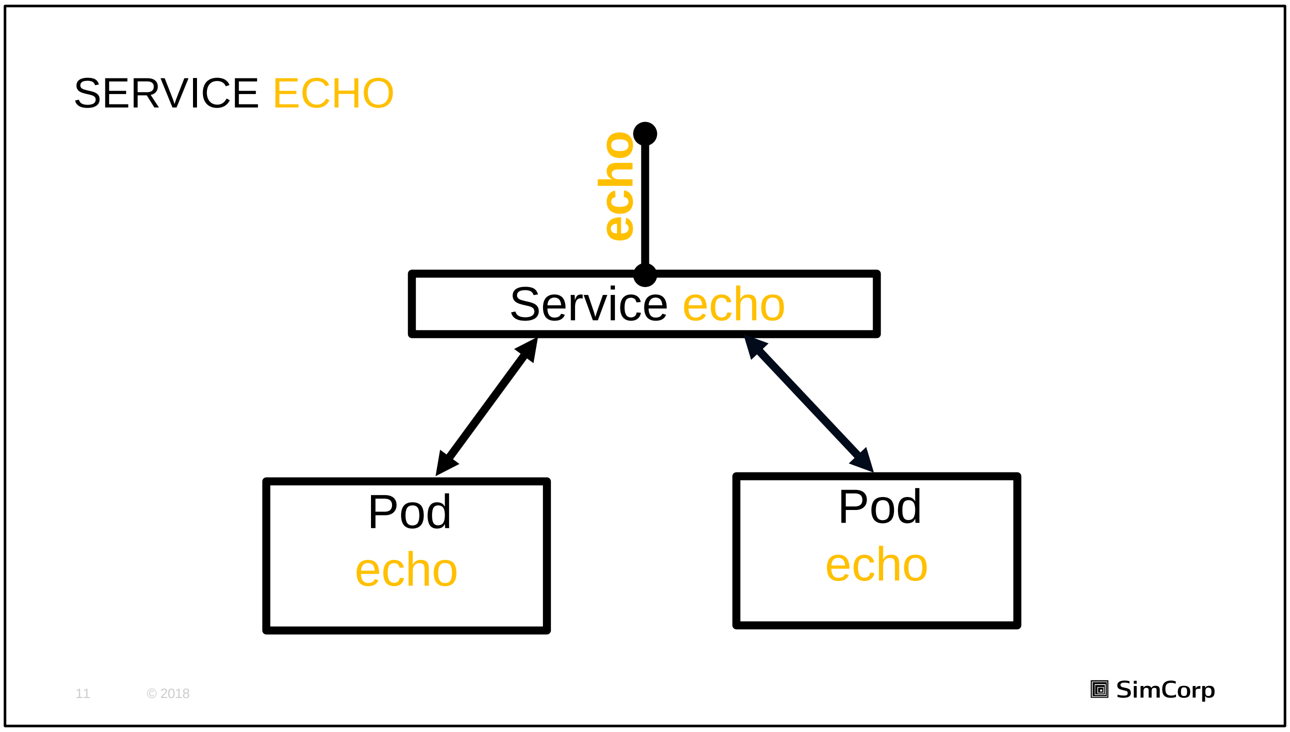
To run this in Kubernetes are needed:

Pods –is a group of one or more containers (such as Docker containers. Kubernetes Pods are mortal

Replica Sets - is the Replication Controller

Services – is an abstraction which defines a logical set of Pods and a policy by which to access them - sometimes called a micro-service.

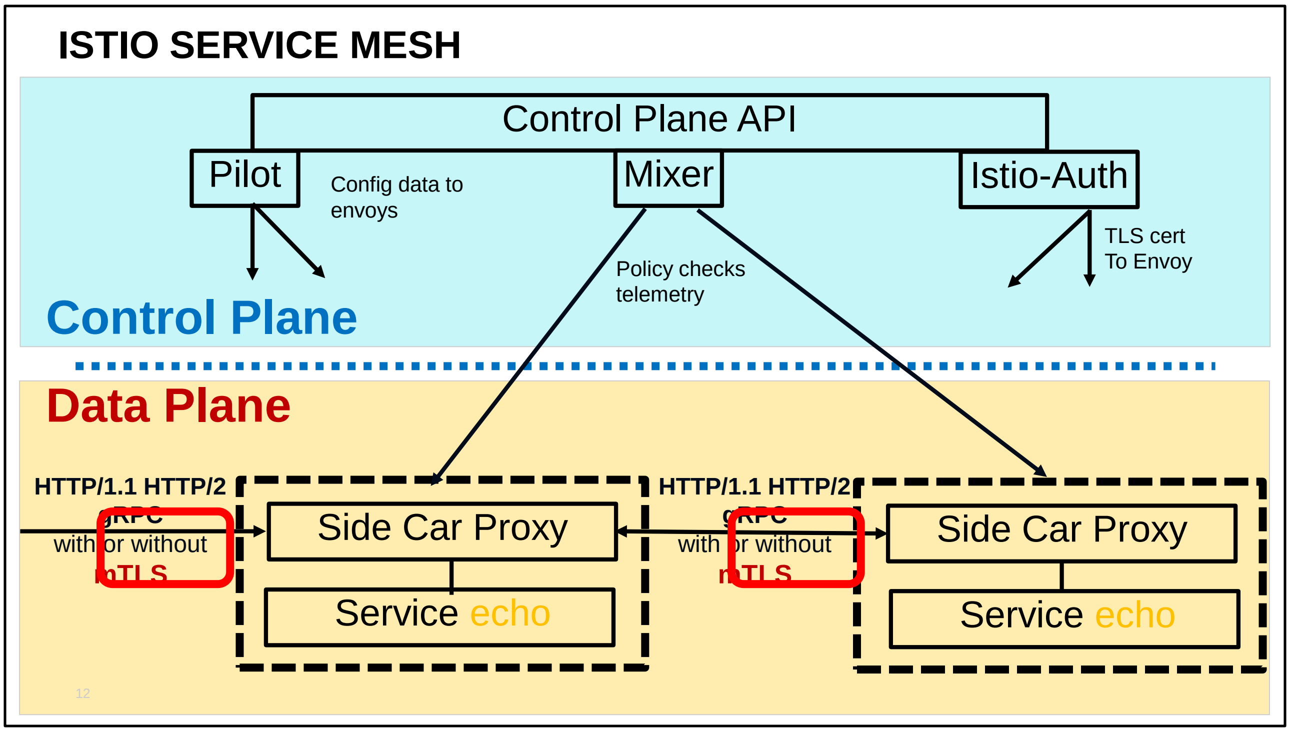
Using **Custom Resource Definition** with Kubeless function.



SERVICE ECHO

Service has fix IP, where Pods are dynamic (scaling)

Pod is like cattle, can die (replaced with new). Kubernetes Pods are mortal



(From https://istio.io/docs/concepts/what-is-istio/)

An Istio service mesh is logically split into a data plane and a control plane.

The **data plane** is composed of a collection of intelligent proxies (Envoys) deployed as sidecars that mediate and control all network communication between microservices.

The **control plane** is used to manage and configure the proxies to route traffic, and enforce polices at the runtime.

Point on Service echos and side cars. In side is used just simple HTTP, it is isolated

Interesting about about Side Car proxy:

- Circut-Breaker pattern.
- mTLS HTTPS encryption is provided by Side Car Proxy

- …

Service mesh is not part of demos.

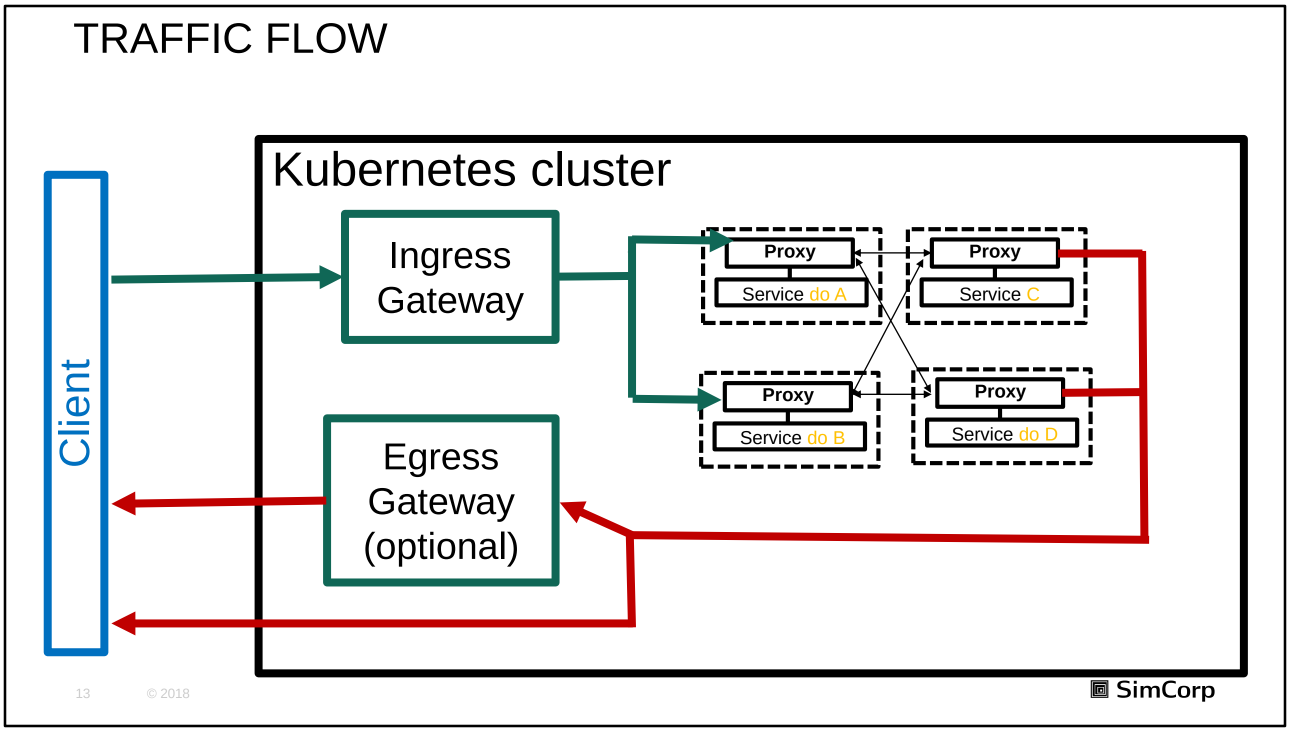
An Istio is mainly composed of the following components:

Envoy: The Envoy is used to mediate all the inbound and outbound traffic for all the services in the service mesh. Functions such as dynamic service discovery, Server Load Balancer, fault injection, and traffic management are supported. The Envoy is deployed as a sidecar to the pods of related services.

Pilot: The Pilot is used to collect and verify the configurations and distribute the configurations to all kinds of Istio components.

Mixer: The Mixer is used to enforce the access control and usage policies in the service mesh, and collect telemetry data from Envoy proxies and other services.

Istio-Auth: Istio-Auth provides strong service-to-service and end user authentication.



An API object that manages external access to the services in a cluster, typically HTTP.

Ingress can provide load balancing, SSL termination and name-based virtual hosting from client to cluster.

Egress other direction i.e. from cluster.

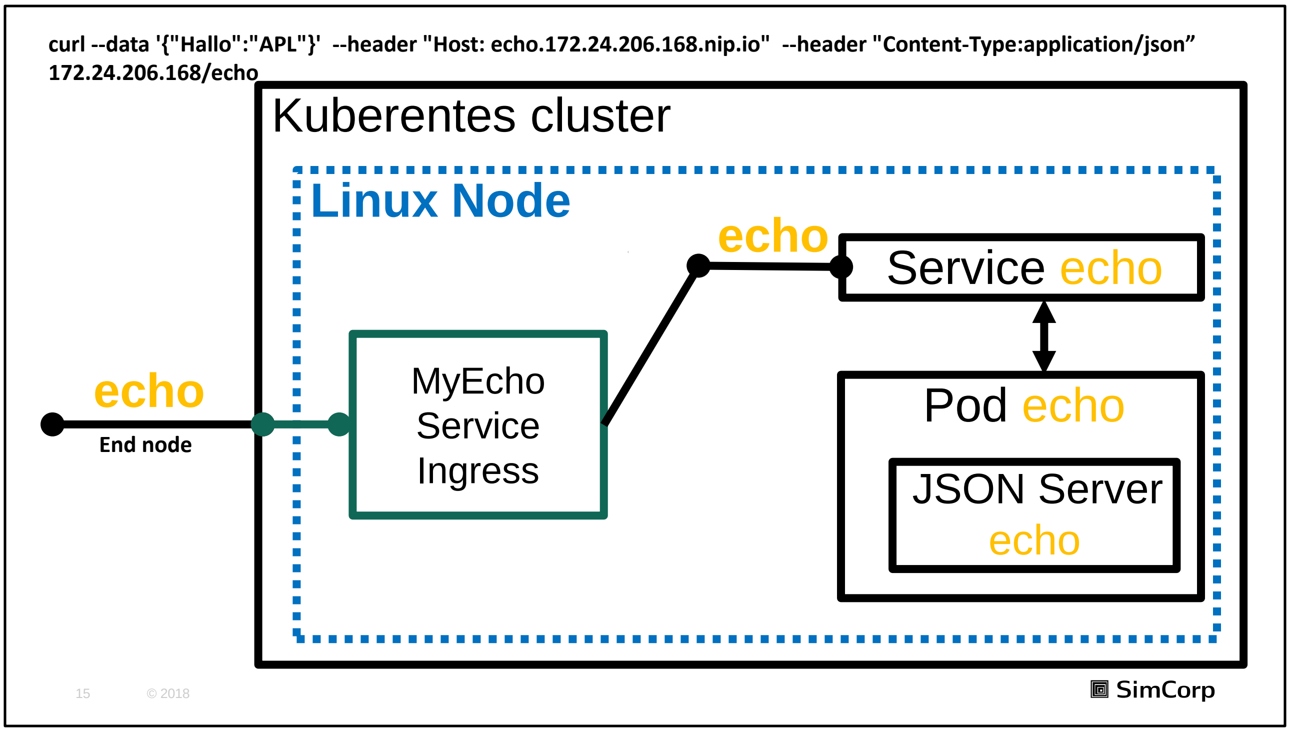
https://istio.io/docs/concepts/traffic-management/

DEMO – HTTP TRIGGER HELLO WORLD APL

- Presenting **HTTP Trigger** from

https://github.com/mvranic/kubeless-apl-demo

SimCorp



```
curl --data '{"Hallo":"APL"}'  --header "Host: echo.172.24.206.168.nip.io"  --header "Content-Type:application/json"
172.24.206.168/echo
```

15 © 2018 SimCorp

Kubernetes cluster in minikube

Endpoint provided by ingress

It run is Linux Node (There is Linux VM behind)

We hit first ingress service

The Echo service

Which know which pod should be used.

Then it is executed APL code in JSON server.

Straight forward and simple.

MATRYOSHKA

BABUSHKA

Source: https://commons.wikimedia.org/wiki/File:Matryoshka_transparent.png

16 © 2018

SimCorp

If Docker is made in Russia, they would call it Matryoshka.

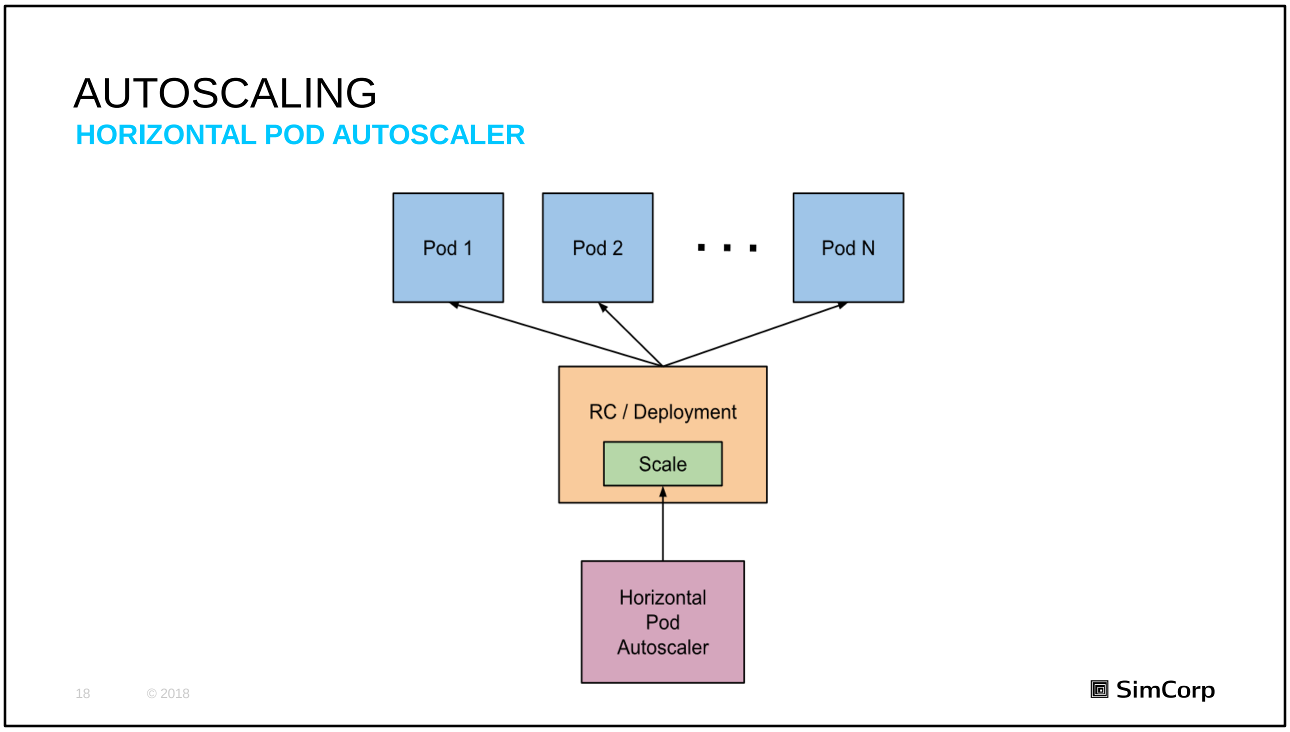
DEMO – PERFORMANCE TEST

- ~4ms for echo invocation

- Presenting **Performance Test** from

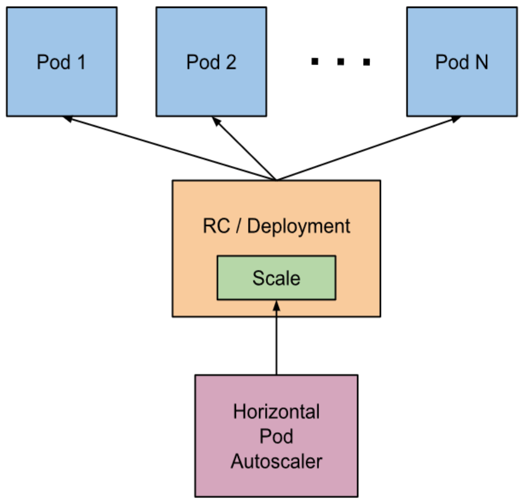
https://github.com/mvranic/kubeless-apl-demo

SimCorp



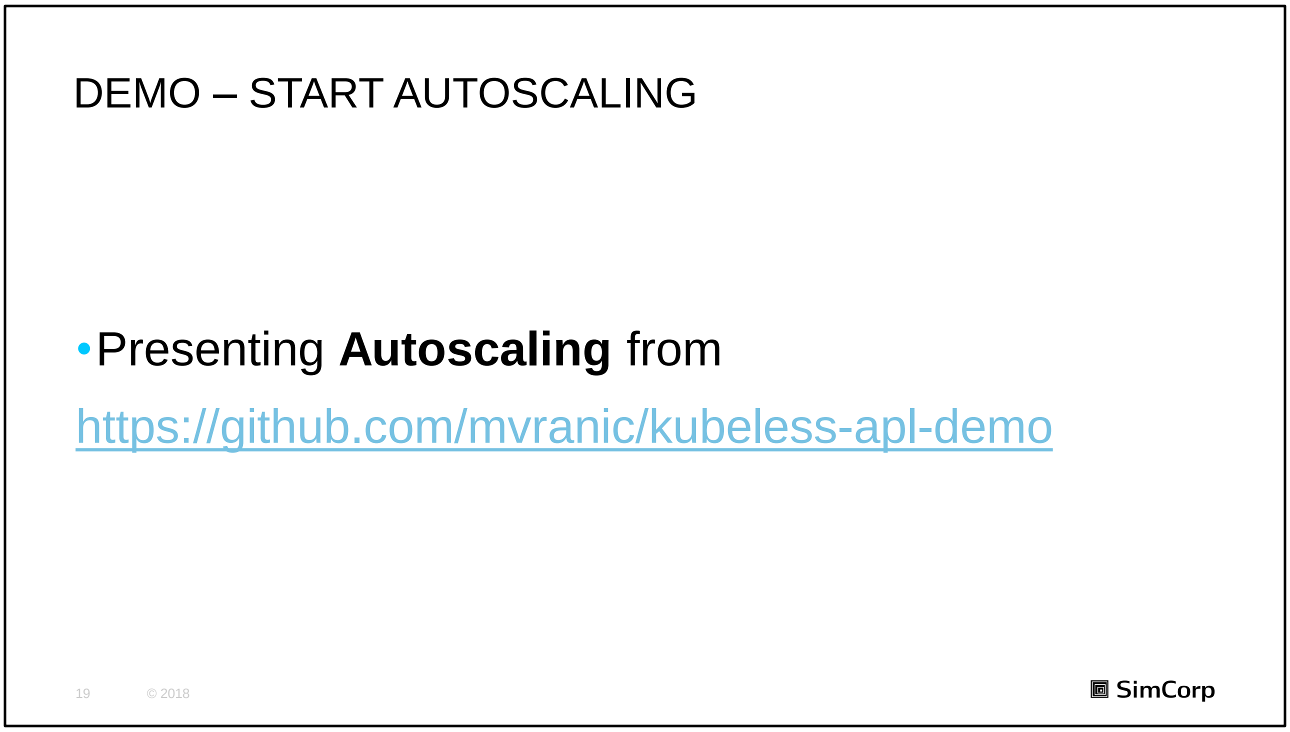
AUTOSCALING

HORIZONTAL POD AUTOSCALER



SimCorp

HPA looks how much CPUs is used and increases or decreases number of pods.



DEMO – START AUTOSCALING

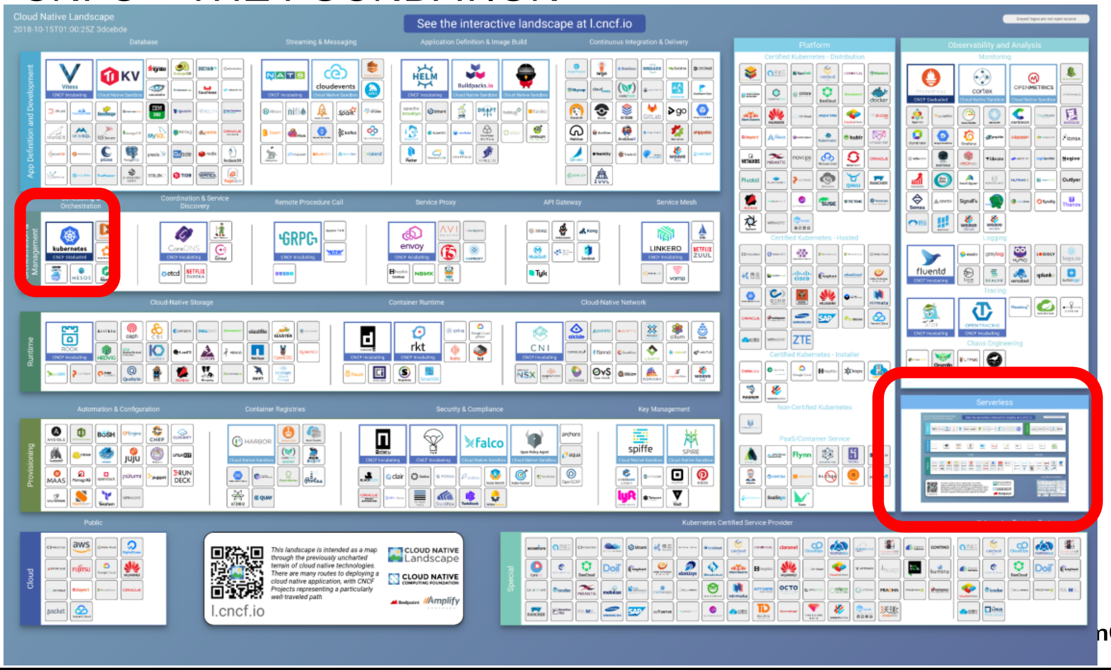
- Presenting **Autoscaling** from

https://github.com/mvranic/kubeless-apl-demo

SimCorp

Continue and show results latter.

CNFC – THE FOUNDATION



nCorp

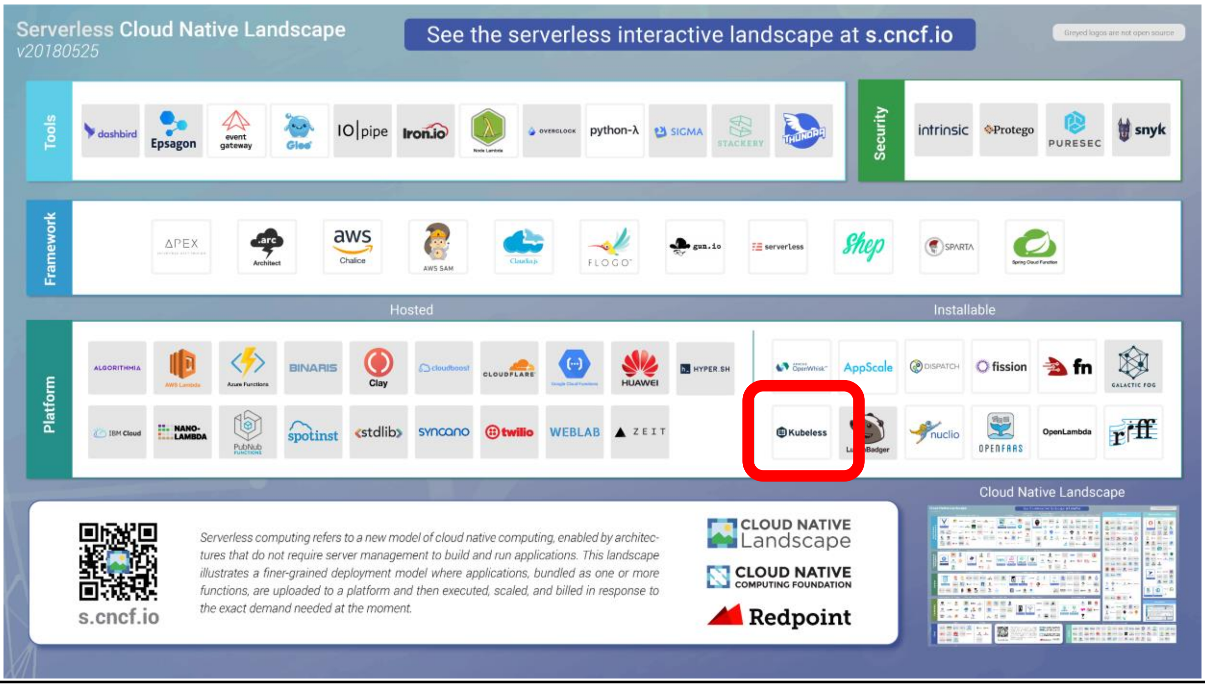
Challenge is to find technology which is appropriate for your business from here.

CNCF is an open source software foundation dedicated to making *cloud* native computing universal and sustainable.

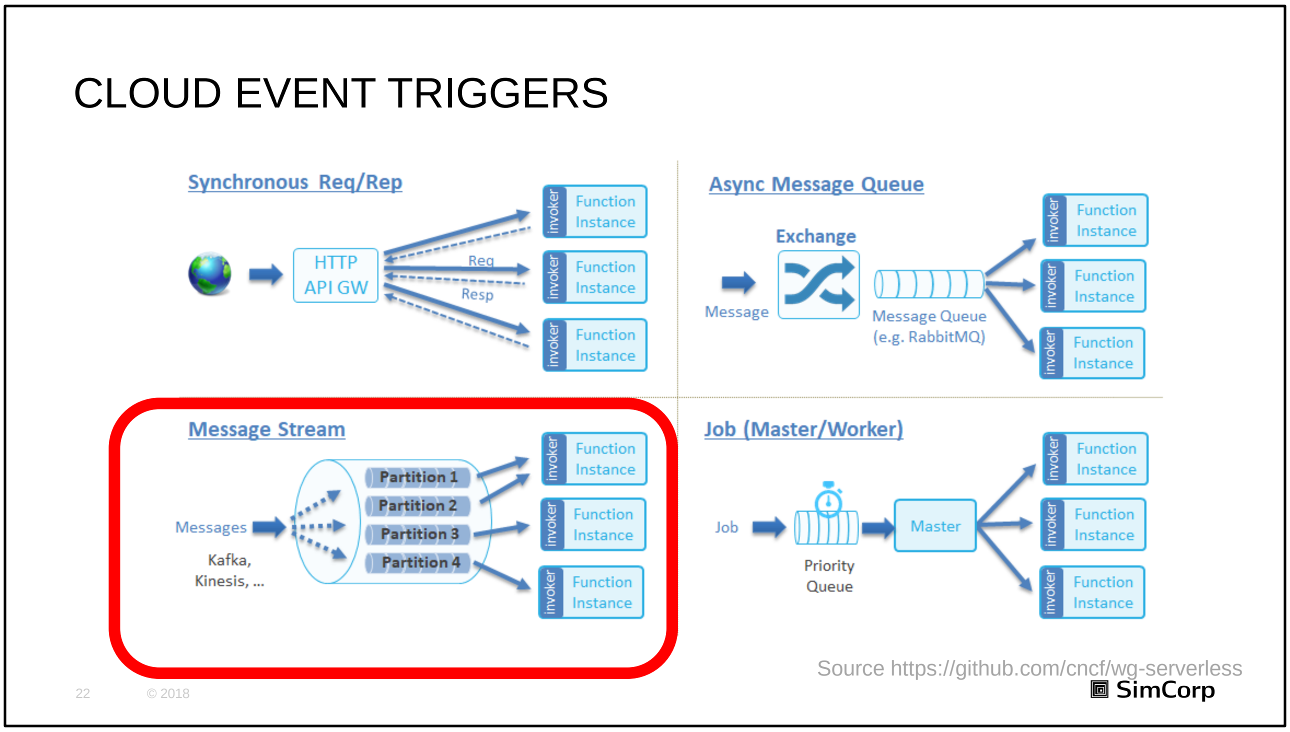
[Cloud Native Computing Foundation: Home Page](https://www.cncf.io/)

https://www.cncf.io/

CNCF SERVERLESS WG



Serverless computing refers to a new model of cloud native computing, enabled by architectures that do not require server management to build and run applications. This landscape illustrates a finer-grained deployment model where applications, bundled as one or more functions, are uploaded to a platform and then executed, scaled, and billed in response to the exact demand needed at the moment.


CLOUD EVENT TRIGGERS

Source https://github.com/cncf/wg-serverless

Serverless is about events and how events are managed.

(From https://github.com/cncf/wg-serverless)

Function Invocation Types

Synchronous Request (Req/Rep), e.g. HTTP Request, gRPC call
Client issues a request and waits for an immediate response.
This is a blocking call.

Asynchronous Message Queue Request (Pub/Sub), e.g. RabbitMQ, AWS SNS, MQTT, Email, Object (S3) change, scheduled events like CRON jobs Messages are published to an exchange and distributed to subscribers

No strict message ordering. Exactly once processing

Message/Record Streams: e.g. Kafka, AWS Kinesis, AWS DynamoDB Streams, Database CDC

An ordered set of messages/records (must be processed sequentially)

• Usually a stream is sharded to multiple partitions/shards with a single worker (the shard consumer) per shard

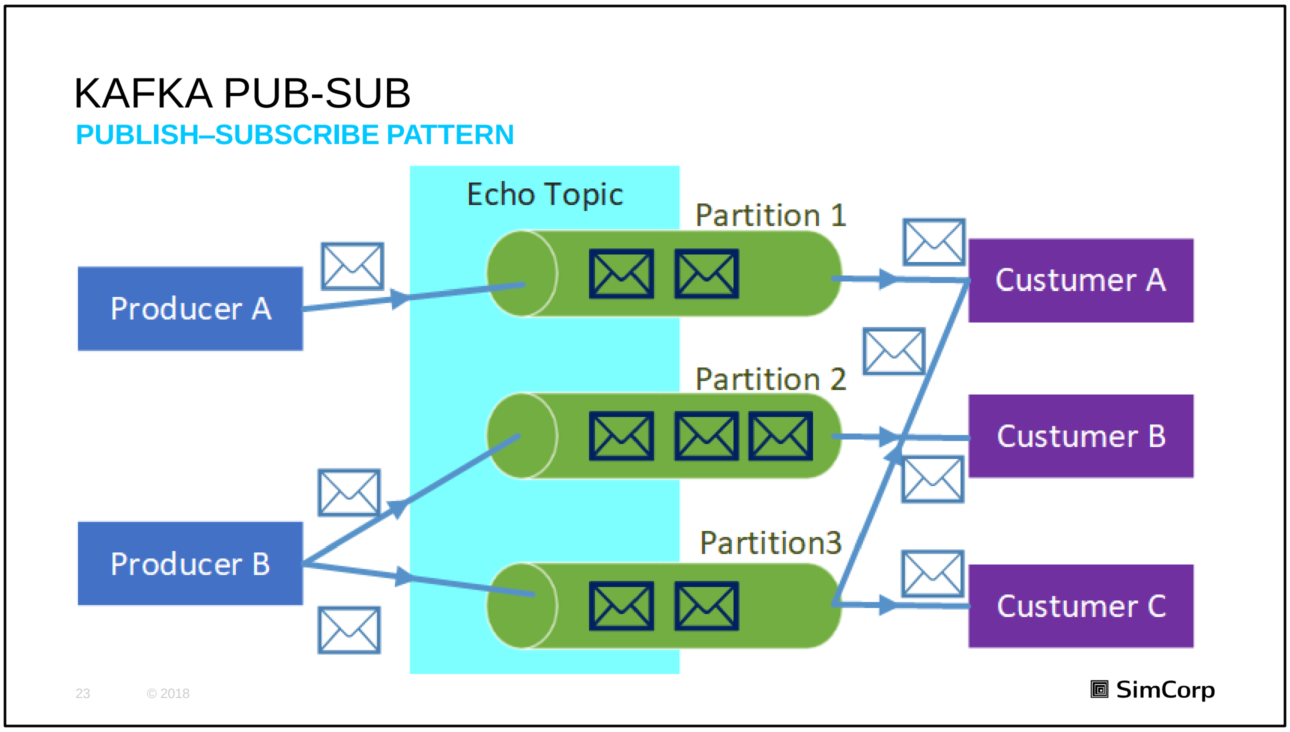
• Stream can be produced from messages, database updates (journal), or files (e.g. CSV, Json, Parquet)

• Events can be pushed into the function runtime or pulled by the function runtime

Batch Jobs, e.g. ETL jobs, distributed deep learning, HPC simulation

• Jobs are scheduled or submitted to a queue, and processed at run time using multiple function instances in parallel, each handling one or more portion of the working set (a task)

• The job is complete when all the parallel workers successfully completed all the computation tasks



KAFKA PUB-SUB

PUBLISH–SUBSCRIBE PATTERN

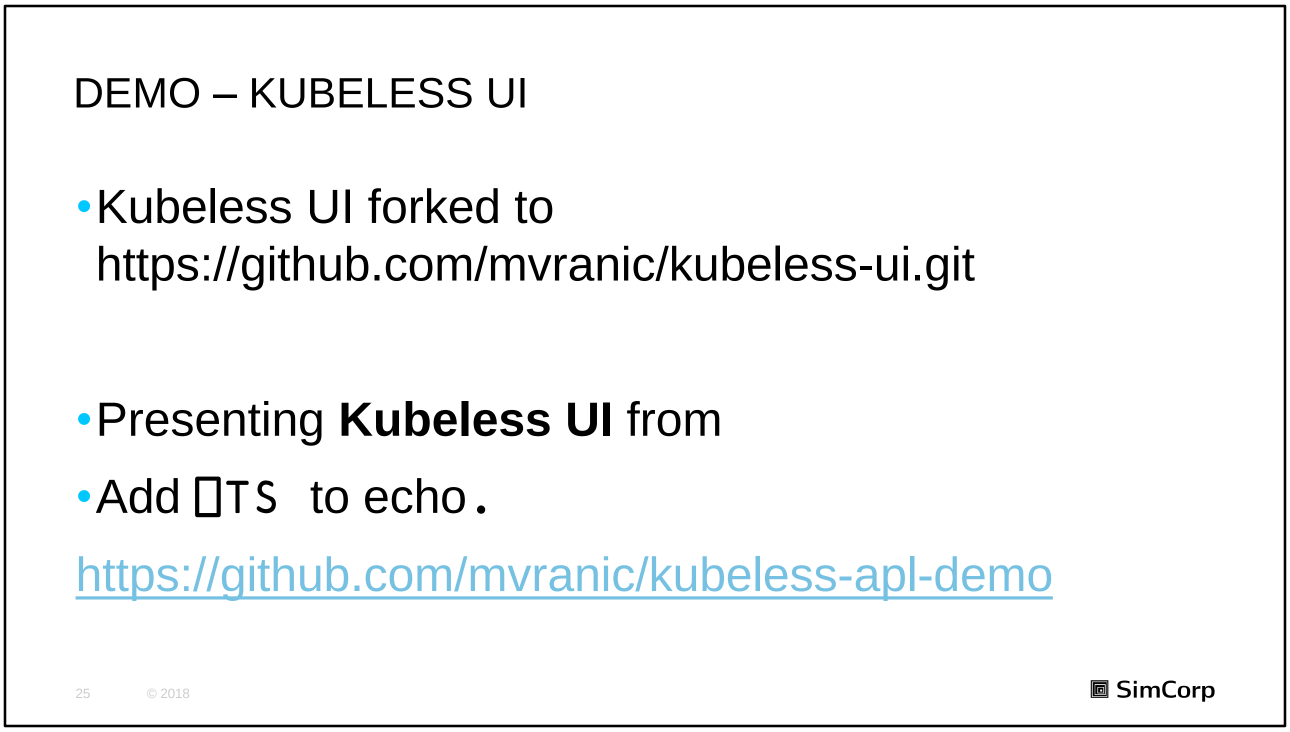
The PubSub function is expected to consume input messages from a predefined topic from a messaging system.

https://kubeless.io/docs/pubsub-functions/

In software architecture, **publish–subscribe** is a messaging pattern where senders of messages, called publishers, do not program the messages to be sent directly to specific receivers, called subscribers, but instead categorize published messages into classes without knowledge of which subscribers, if any, there may be.

DEMO – RESULTS AUTOSCALING AND KAFKA TRIGGER

- Presenting **Kafka Trigger** from https://github.com/mvranic/kubeless-apl-demo
- Results of autoscaling



DEMO – KUBELESS UI

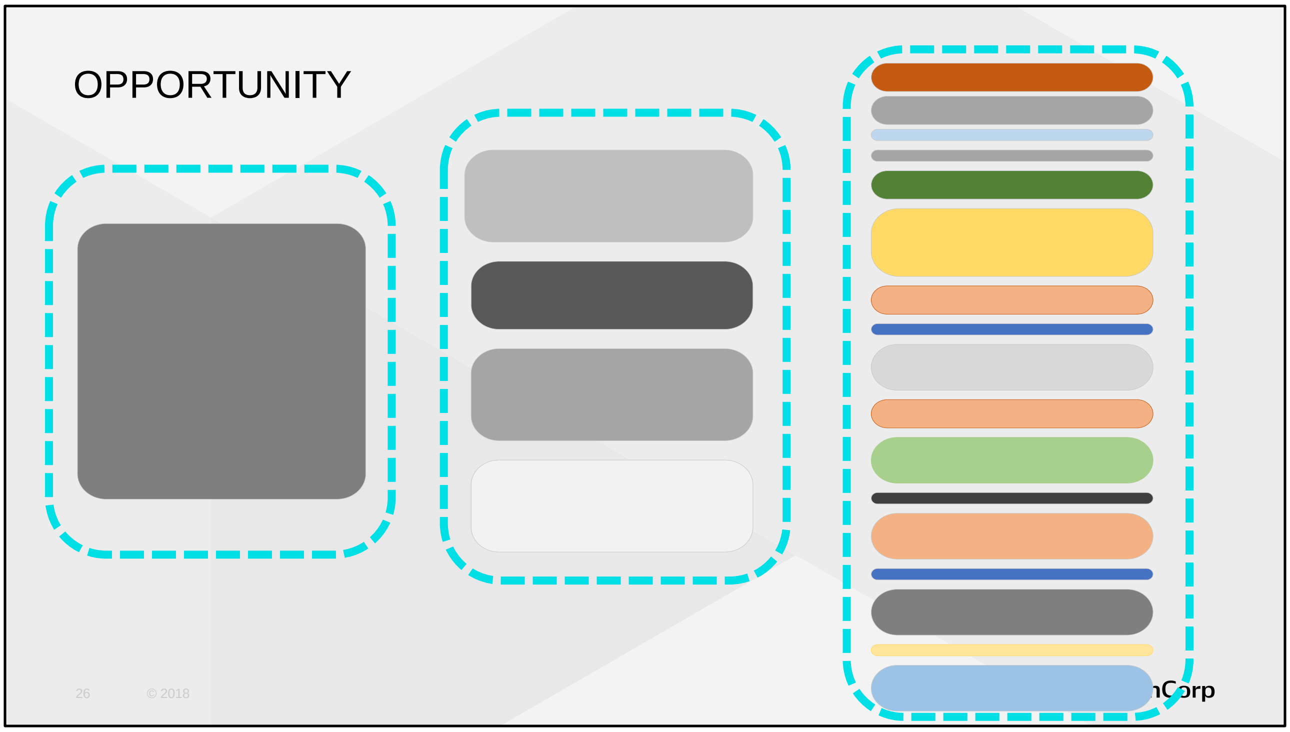
- Kubeless UI forked to https://github.com/mvranic/kubeless-ui.git
- Presenting **Kubeless UI** from
- Add ⎕TS to echo.

https://github.com/mvranic/kubeless-apl-demo

25 © 2018

SimCorp

Extend with ⎕TS for UI.



OPPORTUNITY

26 © 2018

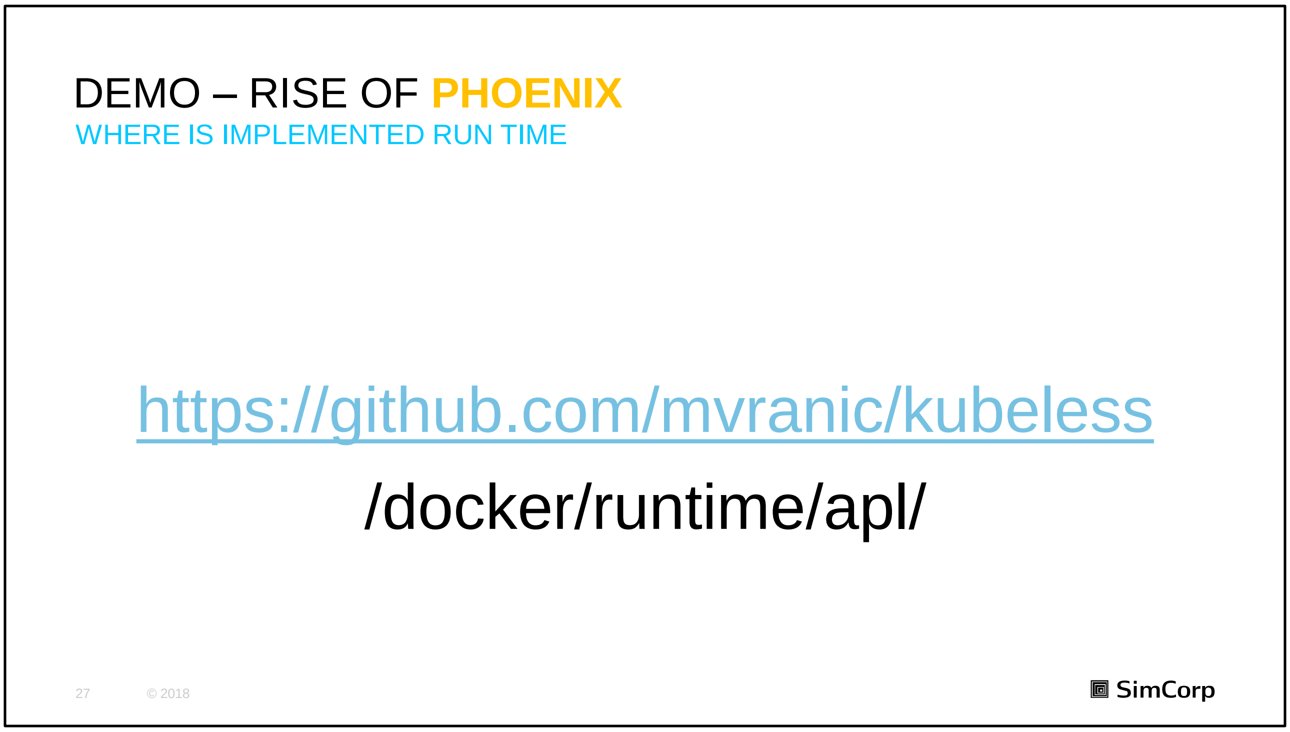
nCorp

Gray is sliver or better to say gold. The system which works and make money.

Architecture:

- Monolith
- Services
- Microservices and serverless

➔ Scaling both in functionalities (business) and execution (performance).



DEMO – RISE OF PHOENIX

WHERE IS IMPLEMENTED RUN TIME

https://github.com/mvranic/kubeless

/docker/runtime/apl/

SimCorp

Show APL with other languages

➔Rise of phoenix

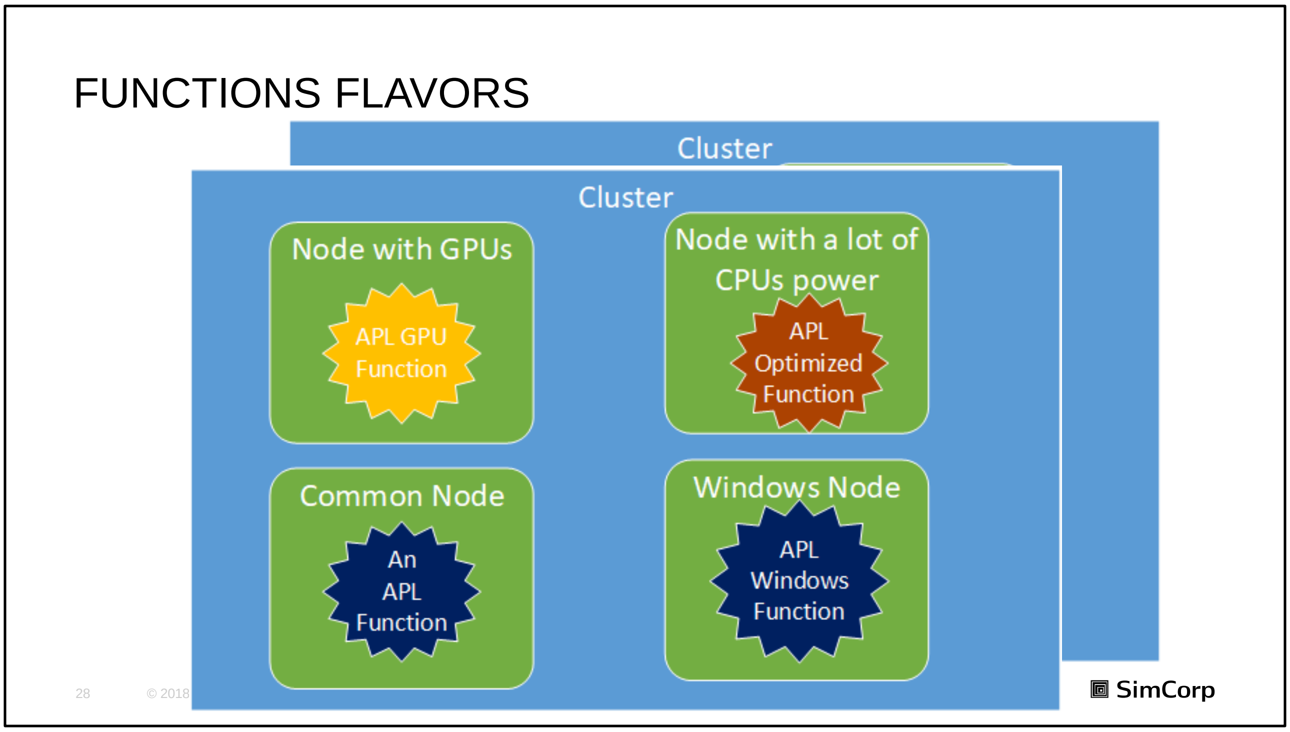
➔ Maybe I should move to marketing

Show: On Github

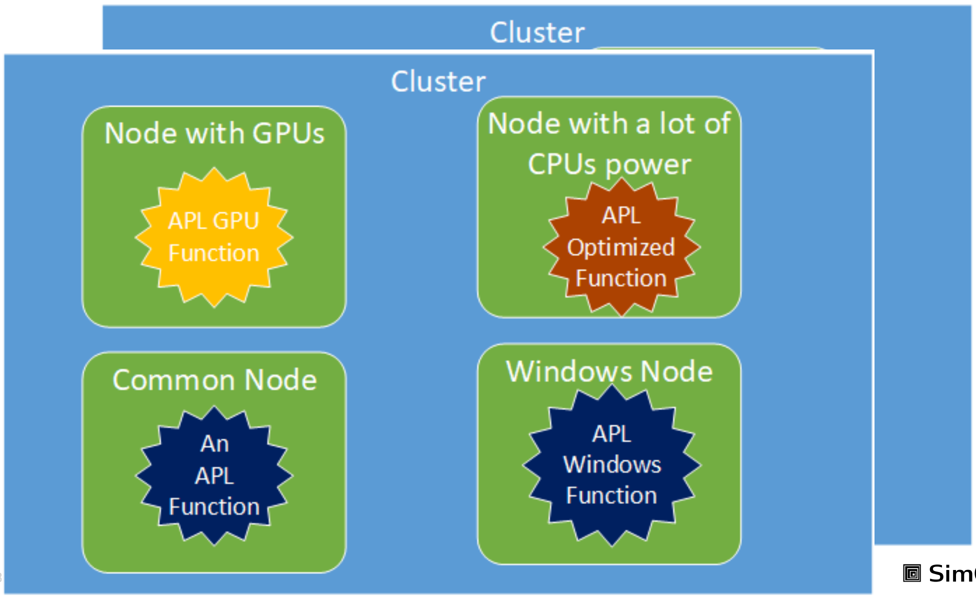
Docker File

kubelessapl/kubeless.dyalog

aplcode/healthz.dyalog



FUNCTIONS FLAVORS



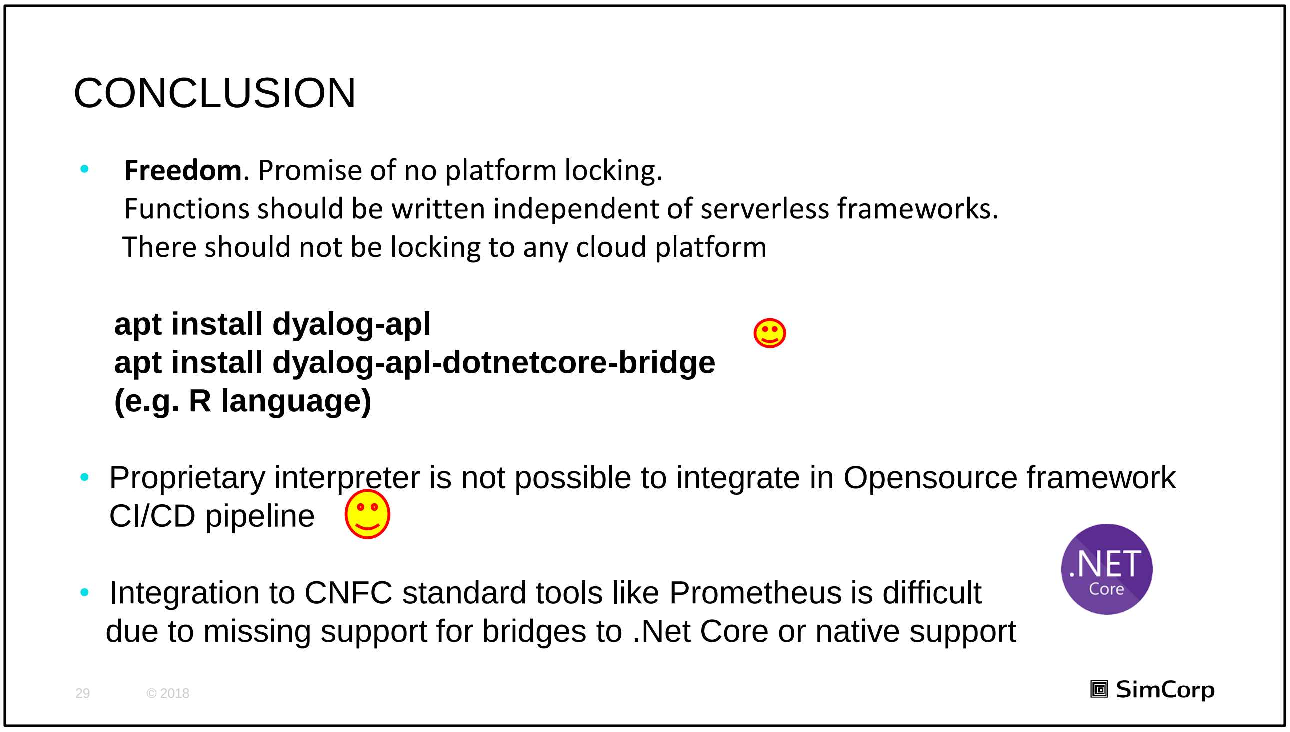
SimCorp

Different APL functions flavors

And they will run with C#, GO, Java Functions.

Labels distinguish where the function will be deployed and executed.

Scaling can be executed over several clusters.



CONCLUSION

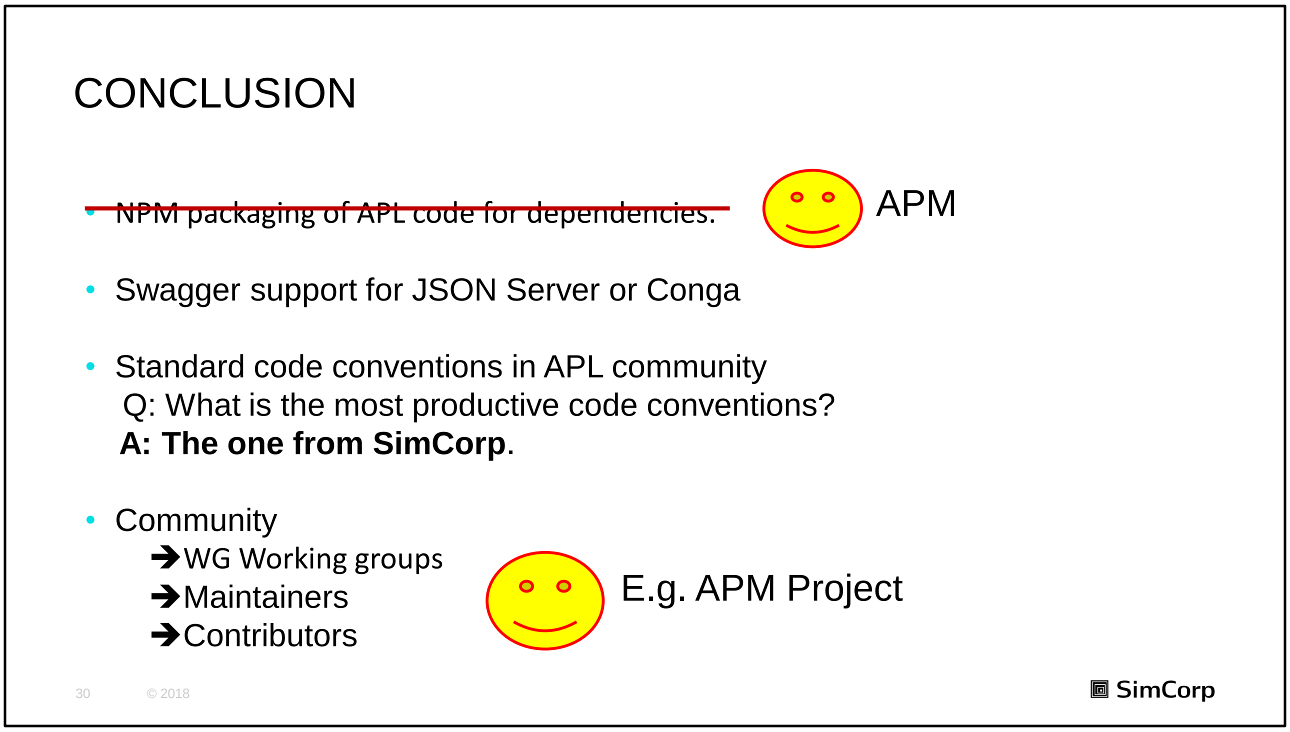
- **Freedom**. Promise of no platform locking.
  Functions should be written independent of serverless frameworks.
  There should not be locking to any cloud platform

  apt install dyalog-apl
  apt install dyalog-apl-dotnetcore-bridge
  (e.g. R language)

- Proprietary interpreter is not possible to integrate in Opensource framework CI/CD pipeline

- Integration to CNFC standard tools like Prometheus is difficult due to missing support for bridges to .Net Core or native support

29 © 2018 SimCorp



CONCLUSION

- ~~NPM packaging of APL code for dependencies.~~ APM
- Swagger support for JSON Server or Conga
- Standard code conventions in APL community
  Q: What is the most productive code conventions?
  A: The one from SimCorp.
- Community
  ➔WG Working groups
  ➔Maintainers
  ➔Contributors

E.g. APM Project

SimCorp

NEXT STEP

Try APL serverless from

https://github.com/mvranic/kubeless-apl-demo

https://github.com/mvranic/kubeless

https://github.com/mvranic/kubeless-bundles

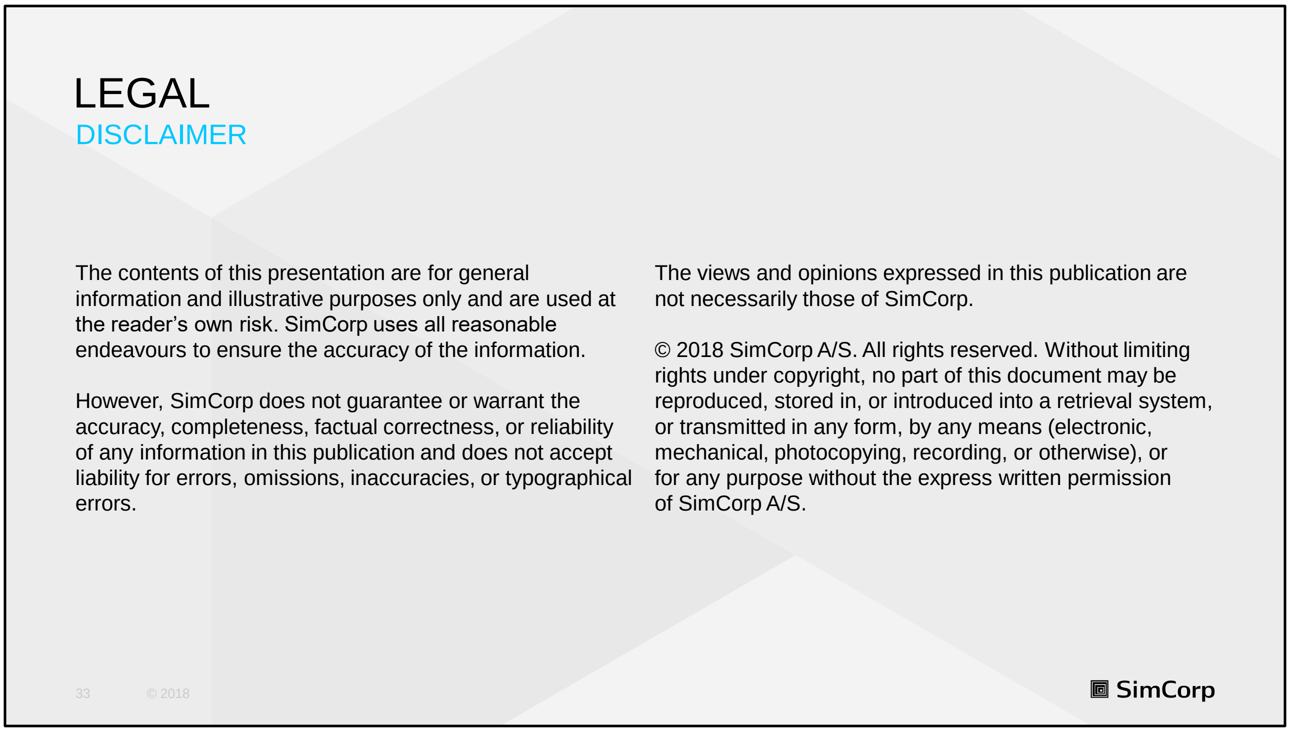
https://github.com/mvranic/kubeless-apl-deployment

https://github.com/mvranic/kubeless-ui

https://github.com/mvranic/JSONServe (used in Kubeless repo.)

SimCorp

SimCorp



LEGAL

DISCLAIMER

33 © 2018

SimCorp