https://github.com/nuclio/nuclio
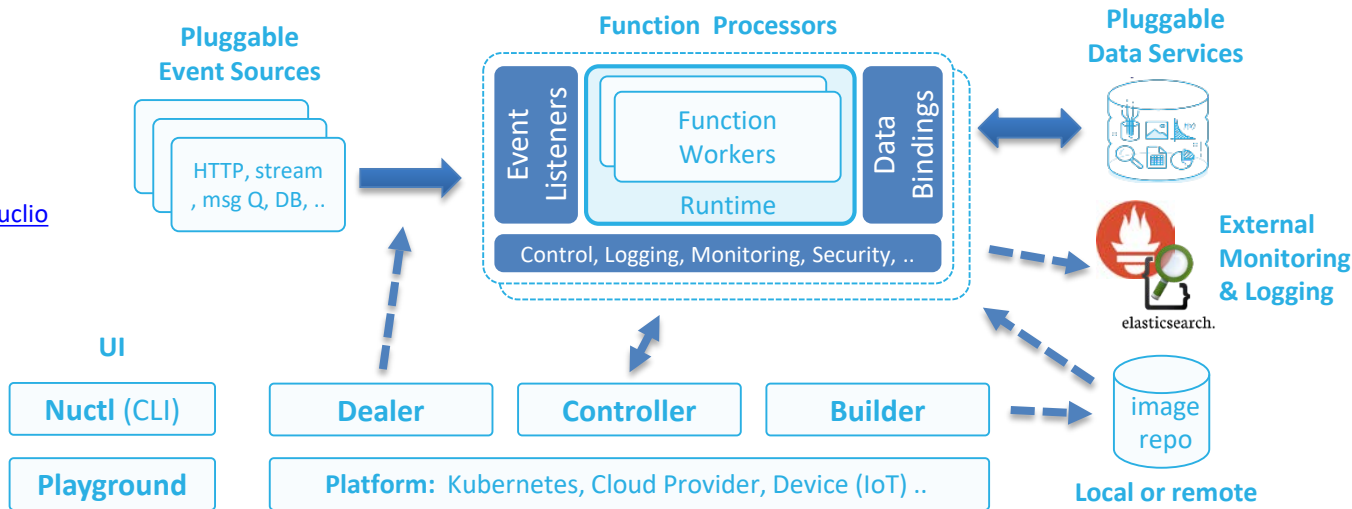
Serverless for Real-Time Events and Data Processing

nuclio - Comprehensive, Open, Portable, & Super Fast "serverless"

https://github.com/nuclio/nuclio

Pluggable Event Sources

HTTP, stream , msg Q, DB, ..

Function Processors

Event Listeners

Function Workers

Data Bindings

Runtime

Control, Logging, Monitoring, Security, ..

Pluggable Data Services

External Monitoring & Logging

elasticsearch.

image repo

Local or remote

UI

Nuctl (CLI)

Playground

Dealer

Controller

Builder

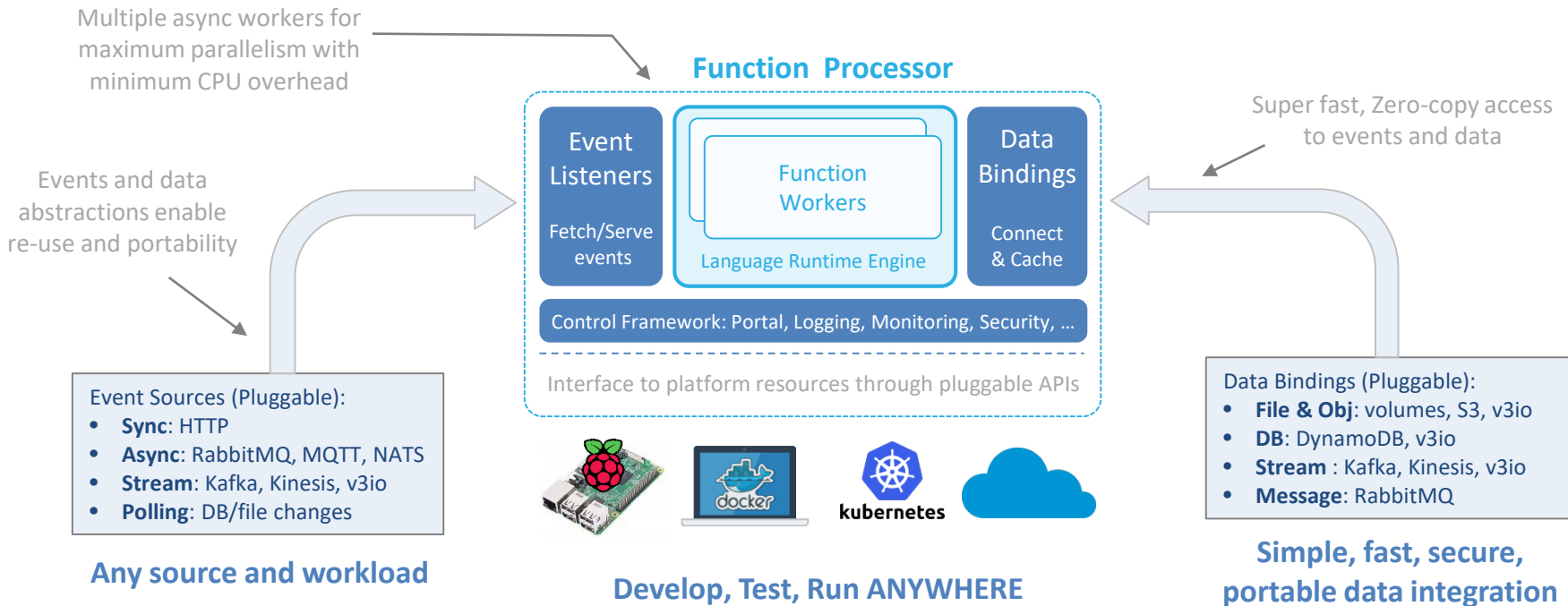**Platform:** Kubernetes, Cloud Provider, Device (IoT) ..

- Real-time processing, low CPU overhead and maximum parallelism
- Simple debugging, regression, and multi-versioned CI/CD pipeline
- Pluggable data/event sources with common APIs
- Portable across low-power devices, laptops, on-prem and public cloud

iguazio

nuclio processor – Fast, Modular & Extensible

400K events/sec per process (100x faster than leading implementations)

Multiple async workers for maximum parallelism with minimum CPU overhead

Events and data abstractions enable re-use and portability

Function Processor

Super fast, Zero-copy access to events and data

Event Listeners

Fetch/Serve events

Function Workers

Language Runtime Engine

Data Bindings

Connect & Cache

Control Framework: Portal, Logging, Monitoring, Security, …

Interface to platform resources through pluggable APIs

Event Sources (Pluggable):
- **Sync**: HTTP
- **Async**: RabbitMQ, MQTT, NATS
- **Stream**: Kafka, Kinesis, v3io
- **Polling**: DB/file changes

Any source and workload

Develop, Test, Run ANYWHERE

Data Bindings (Pluggable):
- **File & Obj**: volumes, S3, v3io
- **DB**: DynamoDB, v3io
- **Stream** : Kafka, Kinesis, v3io
- **Message**: RabbitMQ

Simple, fast, secure, portable data integration

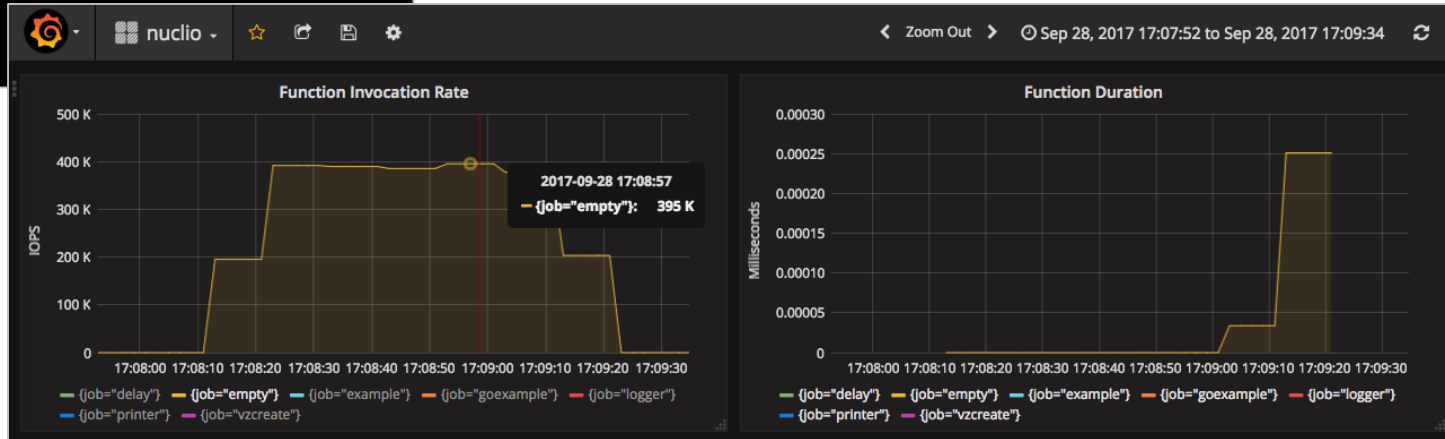# Perf results, single process, using basic function


```
package empty

import (
    "github.com/nuclio/nuclio-sdk"
)

func Empty(context *nuclio.Context, event nuclio.Event) (interface{}, error) {
    return nil, nil
}
```

Tested using:

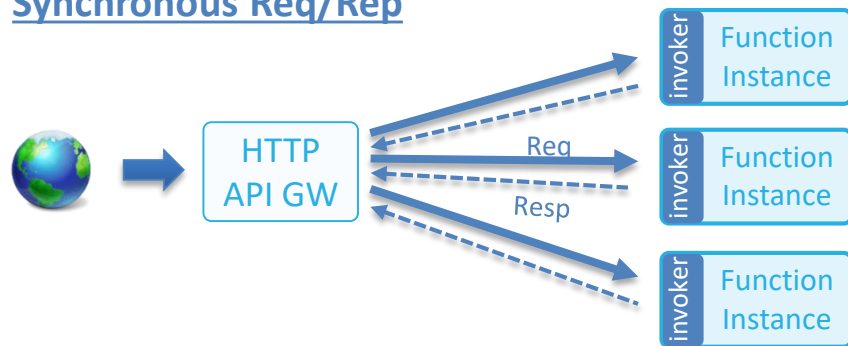https://github.com/v3io/http_blaster
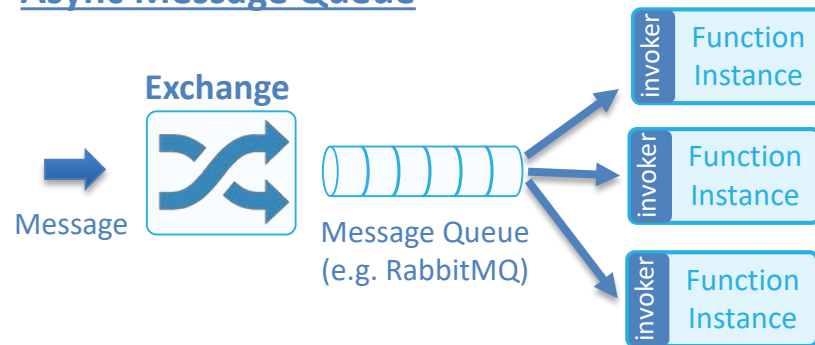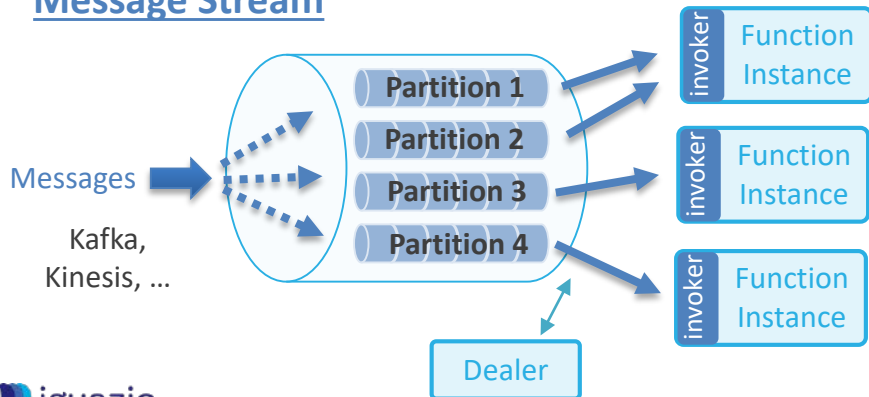
Native
Prometheus
integration

Nuclio invocation modes
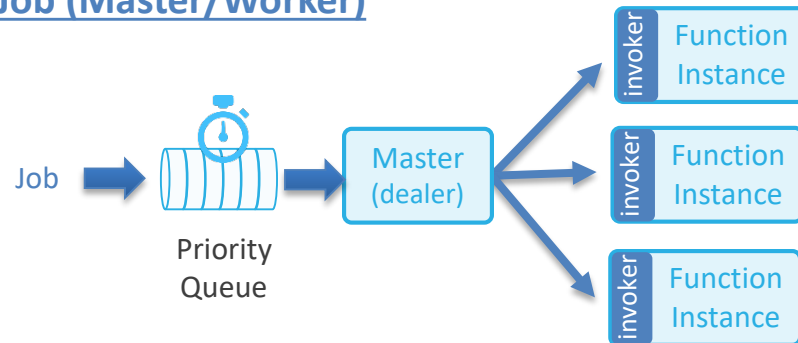
Synchronous Req/Rep

Async Message Queue

Message Stream

Job (Master/Worker)

Nuclio Dealer

- Enable real-time stream processing, batch and interactive jobs on auto-scaling Serverless functions

- By dynamically allocating tasks to workers and handling task lifecycle, checkpoint and completion

Processor

Function Workers

Partitioned data or Stream shards

Processor

Function Workers

Allocate or Re-distribute tasks to processors (1 task per worker)

Dealer

POD Up/Down events

Deployment scale changes

kubernetes

Auto-scale based on CPU load or Q delay

Nuclio controller

Job Spec:
- functions (selector)
- Task num/list
- Max tasks per processor
- Min/Max processors
- ..
- Job Metadata

Job Y (w 4 tasks)   Job X (w 5 tasks)

Every job or stream is partitioned to N smaller Tasks

iguazio

nuclio features & Performance make Serverless broadly applicable
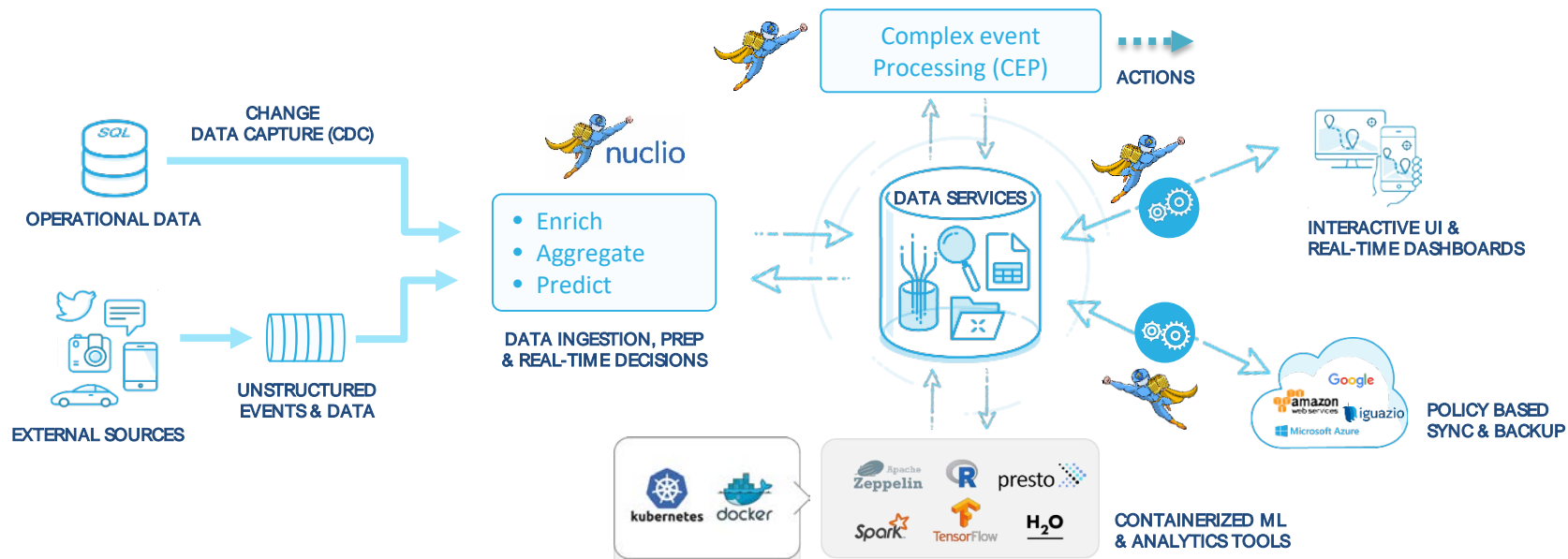
Higher-Productivity | Faster insights | No infrastructure hassle | Lower TCO

Real example: Event Driven Analytics for Connected Cars

* See code in the UI/Playground slide

real-time, auto-Scaling serverless functions

Car state & events
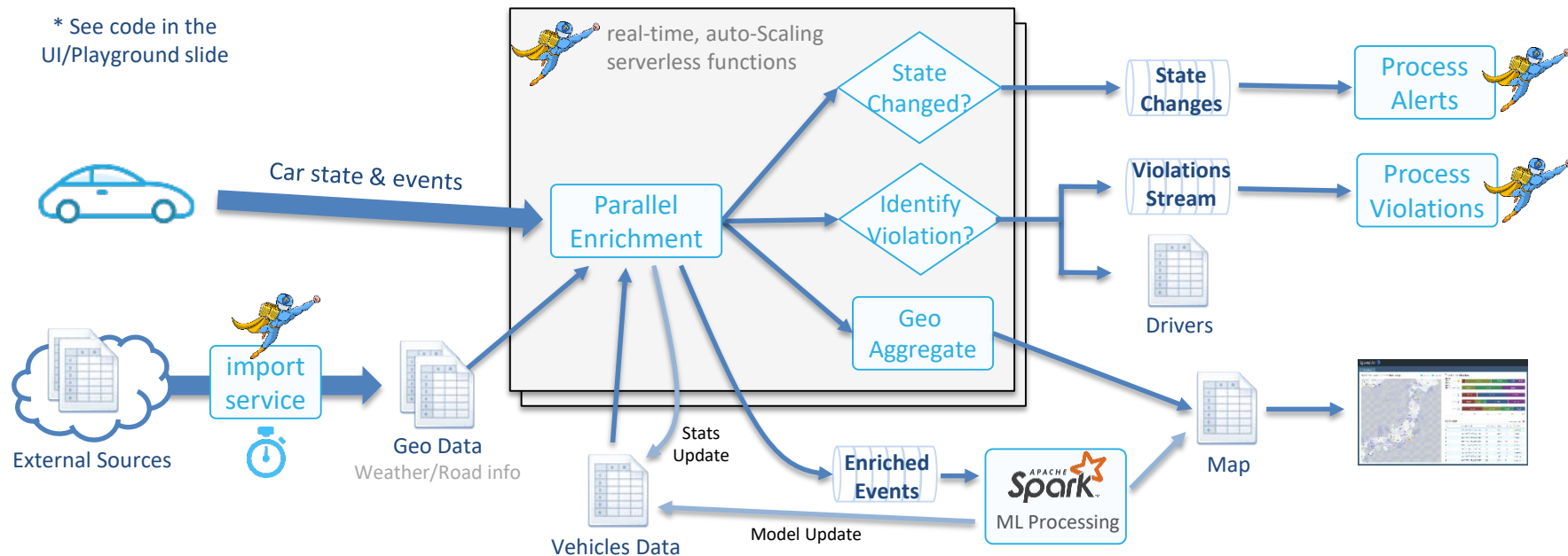
Parallel Enrichment

State Changed? → State Changes → Process Alerts

Identify Violation? → Violations Stream → Process Violations

Drivers

Geo Aggregate

External Sources → import service → Geo Data (Weather/Road info)

Stats Update

Vehicles Data

Enriched Events → APACHE Spark ML Processing

Model Update

Map

Complex Events + Data processed in real-time without the infrastructure hassle

iguazio

iguazio © 2016

nuclio Function Spec

Support Kubernetes CRD:
Functions can be created & deleted using kubectl

*Advanced build instructions & dependencies are in the build.yaml file

```yaml
apiVersion: "nuclio.io/v1"
kind: Function
metadata:
  name: example
  namespace: myproject
  labels:
    author: joe
spec:
  image: example:latest
  replicas: 0
  maxReplicas: 10
  env:
  - name: SOME_ENV
    value: abc
  - name: SECRET_PASSWORD_ENV_VAR
    valueFrom:
      secretKeyRef:
        name: my-secret
        key: password
  resources:
    requests:
      memory: "64Mi"
      cpu: "250m"
    limits:
      memory: "128Mi"
      cpu: "500m"
  dataBindings:
    db0:
      class: v3io
      secret: mysecret
      url: http://199.19.70.139:8081/1024
```

namespaced

tags/labels used for search and event sources (Label Selectors)

Various src code options*: inline code, path (local/http/git), or local/remote pre-built image

Control Min/Max Replicas for controlled auto-scale

Pass text or secret environment variables (k8s convention)

Flex resource allocation, GPUs are coming

Pluggable Data Sources

Nuclio common event model

```go
type Event interface {
    // Unique ID of the event
    GetID() string
    // Event Source class, kind, ver, schema
    GetEventSource() SourceInfoProvider
    // Event Source address (e.g. origin host IP:port, origin stream, ..)
    GetSourceAddress() string
    // Source identity (e.g. authenticated by a gateway)
    GetSourceIdentity() string
    // Content type e.g. application/json
    GetContentType() string
    // byte array of content, encoding defined by content type
    GetBody() []byte
    // Get header(s) (e.g. HTTP, AMQP, or anything injected by the source)
    // also have convenience methods: GetHeaderByteSlice, GetHeaderString
    GetHeader(key string) interface{}
    GetHeaders() map[string]interface{}
    // Get field(s), decode fields in the body (e.g. json, DB record, ..)
    // Allow functions to ignore the specific event encoding, e.g. emulate DB record via HTTP
    // also have convenience methods: GetFieldByteSlice, GetFieldString, GetFieldInt
    GetField(key string) interface{}
    GetFields() map[string]interface{}
    // Original event timestamp or gateway timestamp (if origin timestamp not specified)
    GetTimestamp() time.Time
    // Logical path requested by the event (e.g. HTTP request path, stream name, etc.)
    GetPath() string
    // URL object for HTTP requests, for convenience
    GetURL() URL
    // HTTP or transport method
    GetMethod() string
    // Translate event to a json byte array
    AsJson() []byte
}
```

Simplify and generalize client implementation

Enable zero copy and zero ser/des when possible

iguazio

Context.logger Interface

```
type Logger interface {

    // emit a log entry of a given verbosity. the first argument may be an object, a string
    // or a format string. in case of the latter, the following varargs are passed
    // to a formatter (e.g. fmt.Sprintf)
    Error(format interface{}, vars ...interface{})
    Warn(format interface{}, vars ...interface{})
    Info(format interface{}, vars ...interface{})
    Debug(format interface{}, vars ...interface{})

    // emit a structured log entry. example:
    //
    // 1.InfoWith("The message",
    //      "first-key", "first-value",
    //      "second-key", 2)
    //
    ErrorWith(format interface{}, vars ...interface{})
    WarnWith(format interface{}, vars ...interface{})
    InfoWith(format interface{}, vars ...interface{})
    DebugWith(format interface{}, vars ...interface{})

    // flushes buffered logs, if applicable
    Flush()

    // returns a child logger, if underlying logger supports hierarchal logging
    GetChild(name string) interface{}
}
```
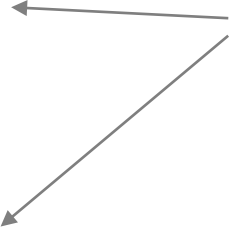
Support both structured & unstructured logging

Support nested/hierarchical logs

One log interface, multiple implementations (screen, file, stream, http, ..), extensible

iguazio

Default Context.DataBinding API (sync & async ver), can be overwritten

Service	Major APIs	Main Request Params

Object
e.g. S3, Minio, v3io
	`ListObjects`
	`GetObject`
	`PutObject`
	`DeleteObject`
		`Bucket, Prefix, MaxKeys`
		`Bucket, Key, Range`
		`Bucket, Key ,Metadata, Body`
		`Bucket, Key`

NoSQL
e.g. DynamoDB,
Cassandra, v3io
	`GetItem`
	`GetItems`
	`PutItem`
	`UpdateItem`
	`DeleteItem`
		`Table, Key ,Projection`
		`Table, ConditionExpression, ProjectionExpression, Limit`
		`Table, Key, ProjectionExpression, item`
		`Table, Key, UpdateExpression, ConditionExpression`
		`Table, Key, ConditionExpression`

Stream
e.g. Kinesis, Kafka,
v3io
	`GetRecords`
	`PutRecords`
	`Seek`
		`Stream, ShardId, Location, Limit`
		`Stream, Records`
		`Stream, ShardId, SeekType, SeekTime, StartingSequence, Timestamp`

File	`Open`
	`Read`
	`Write`
		`Path, Mode, flags`
		`Handle, offset, size`
		`Handle, offset, size, data`

iguazio

Nuclio Playground (run as isolated k8s deployment)

CLI (run command example)

```
$ nuctl run --help
Build, deploy and run a function

Usage:
  nuctl run function-name [flags]

Flags:
      --data string           Comma separated list of data bindings (in json)
      --data-bindings string  JSON encoded data bindings for the function
      --desc string           Function description
  -d, --disabled              Start function disabled (don't run yet)
  -e, --env string            Environment variables (name1=val1,name2=val2..)
      --events string         Comma separated list of event sources (in json)
  -f, --file string           Function Spec File
  -h, --help                  help for run
  -i, --image string          Docker image name, will use function name if not specified
  -l, --labels string         Additional function labels (lbl1=val1,lbl2=val2..)
      --max-replica int32     Maximum number of function replicas
      --min-replica int32     Minimum number of function replicas
      --no-pull               Don't pull base images - use local versions
      --nuclio-src-dir string  Local directory with nuclio sources (avoid cloning)
      --nuclio-src-url string  nuclio sources url for git clone (default "https://github.com/nuclio/nuclio.git")
  -o, --output string         Build output type - docker|binary (default "docker")
  -p, --path string           Function source code path
      --port int32            Public HTTP port (node port)
      --publish               Publish the function
  -r, --registry string       URL of container registry (env: NUCTL_REGISTRY)
      --run-registry string   The registry URL to pull the image from, if differs from -r (env: NUCTL_RUN_REGISTRY)
      --runtime string        Runtime – golang, python, ..
  -s, --scale string          Function scaling (auto|number) (default "1")
      --version string        Docker image version (default "latest")

Global Flags:
  -k, --kubeconfig string     Path to Kubernetes config (admin.conf) (default ~/.kube/config")
  -n, --namespace string      Kubernetes namespace (default "default")
  -v, --verbose               verbose output
```
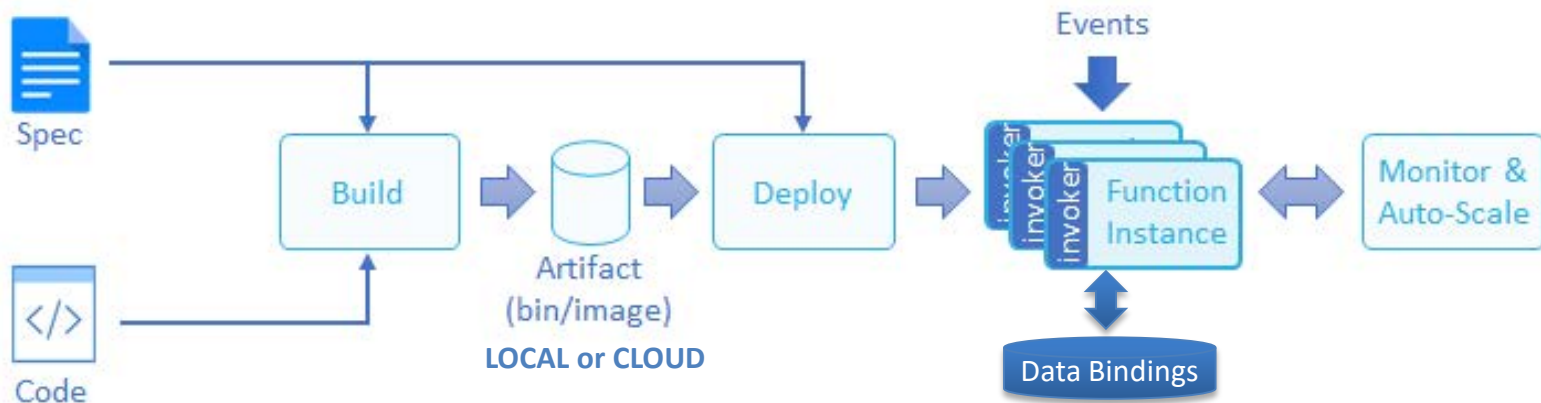
See more in: https://github.com/nuclio/nuclio/blob/master/docs/nuctl/nuctl.md

iguazio

Enabling Simplest and Continuous Dev & Ops (CI/CD)

$ nuctl run <name> <source> [options]

One Click to test, deploy, upgrade or rollback code
Runs ANYWHERE, Self-healing and Auto-Scaling

nuclio

ORCHESTRATION

kubernetes docker

UNIFIED & AUTOMATED MANAGEMENT

SERVERLESS PROCESSING

Event Driven Code

ML & AI FRAMEWORKS

Zeppelin R presto

Spark TensorFlow H2O

DATA SERVICES APIS

← Low latency 100GbE TCP or RDMA Data Fabric (V3IO) →

NoSQL API	Stream API	Object API	File API
Security	Queries & Functions	Unified Data	Data Lifecycle

HYBRID DEPLOYMENT

On-Premises   Hosted Cloud   Edge

- **Used in iguazio platform**
  - Developed for the real world
- **Now completely re-written to:**
  - Support the broader open source & CNCF eco-system
  - Incorporate learnings from G1
  - Future proof architecture
  - Address new use cases

iguazio