



nuclio

<https://github.com/nuclio/nuclio>

<https://www.youtube.com/watch?v=xlOp9BR5xcs>

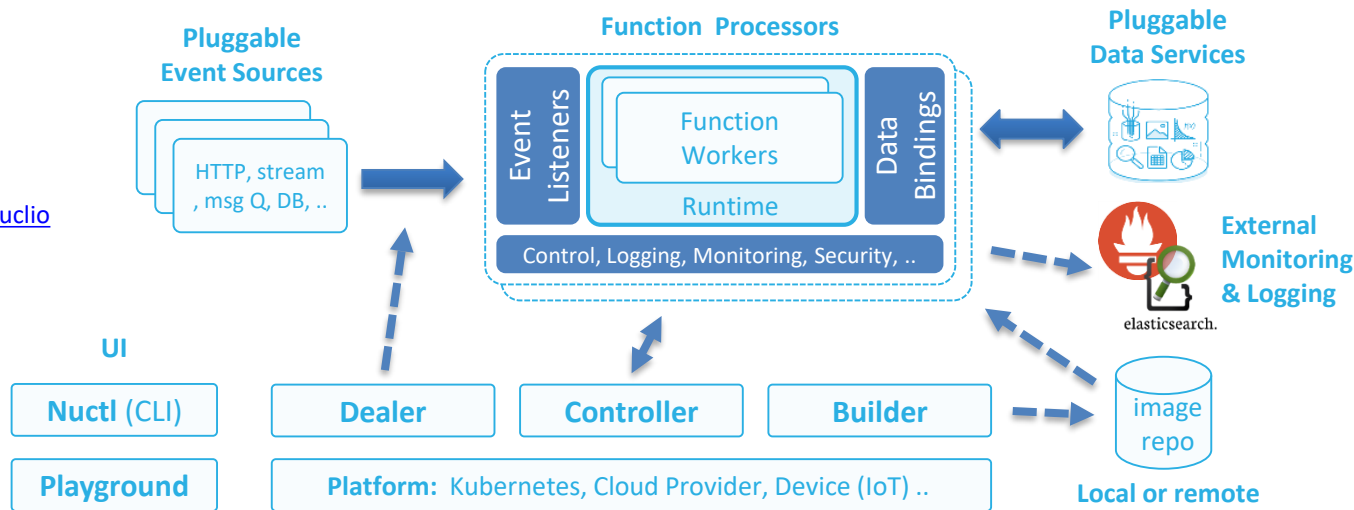
Serverless for Real-Time Events and Data Processing

nuclio - Comprehensive, Open, Portable and Super Fast “Serverless”



nuclio

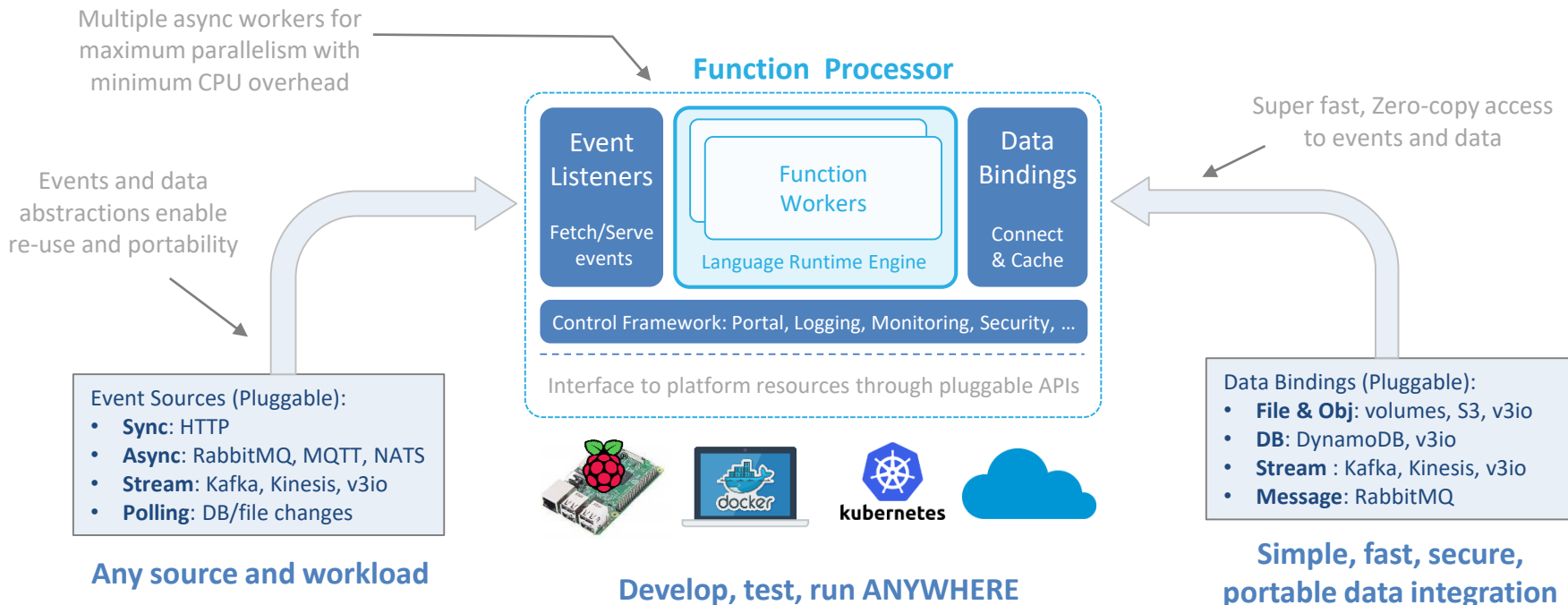
<https://github.com/nuclio/nuclio>



- 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

nuclio Processor – Fast, Modular and Extensible

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



Perf Results, Single Process, Using Basic Functions

```

2017/09/28 17:08:01 Starting http_blaster
2017/09/28 17:08:01 Running test on [1]:30573, tls mode=false, block size=5, test timeout 2m40s
2017/09/28 17:08:01 Adding executor for get
2017/09/28 17:08:01 at executor start {multi get from 20 workers GET 1m0s 0 128 0 map[] 0 0 false performance 0
0 0}
2017/09/28 17:08:01 Wait for executors to finish
2017/09/28 17:09:01 Ending multi get from 20 workers
2017/09/28 17:09:01 report for wL 0:
2017/09/28 17:09:01 Total Requests 23417291
2017/09/28 17:09:01 Min: 0s
2017/09/28 17:09:01 Max: 211.343548ms
2017/09/28 17:09:01 Avg: 1.887384ms
2017/09/28 17:09:01 Error Count: 0
2017/09/28 17:09:01 Statuses:
2017/09/28 17:09:01 503 - 19
2017/09/28 17:09:01 200 - 23417272
2017/09/28 17:09:01 lops: 390288
2017/09/28 17:09:01 status code 200 occurred 99.999919% during the test "multi get from 20 workers"
2017/09/28 17:09:01 report for wL 0:
2017/09/28 17:09:01 Total Requests 23417291
2017/09/28 17:09:01 Min: 0s
2017/09/28 17:09:01 Max: 211.343548ms
2017/09/28 17:09:01 Avg: 1.887384ms
2017/09/28 17:09:01 Error Count: 0
2017/09/28 17:09:01 Statuses:
2017/09/28 17:09:01 200 - 23417272
2017/09/28 17:09:01 503 - 19
2017/09/28 17:09:01 lops: 390288
2017/09/28 17:09:01 Duration: 1m0.476341776s
2017/09/28 17:09:01 Overall Results:
2017/09/28 17:09:01 Overall Requests: 23417291
2017/09/28 17:09:01 Overall GET Requests: 23417291
2017/09/28 17:09:01 Overall GET Min Latency: 0s
2017/09/28 17:09:01 Overall GET Max Latency: 211.343548ms
2017/09/28 17:09:01 Overall GET Avg Latency: 1.887384ms
2017/09/28 17:09:01 Overall PUT Requests: 0
2017/09/28 17:09:01 Overall PUT Min Latency: 0s
2017/09/28 17:09:01 Overall PUT Avg Latency: 0s
2017/09/28 17:09:01 Overall IOPS: 390288
2017/09/28 17:09:01 Overall GET IOPS: 390288
2017/09/28 17:09:01 Overall PUT IOPS: 0

```

```
package empty
```

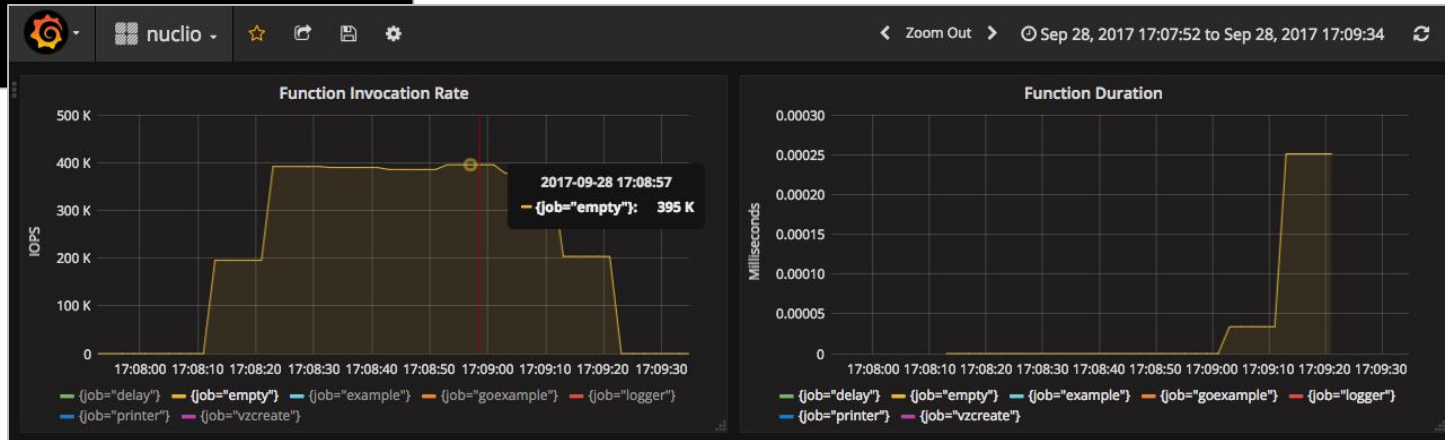
```
import (
    "github.com/nuclio/nuclio-sdk"
)
```

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

Tested using:

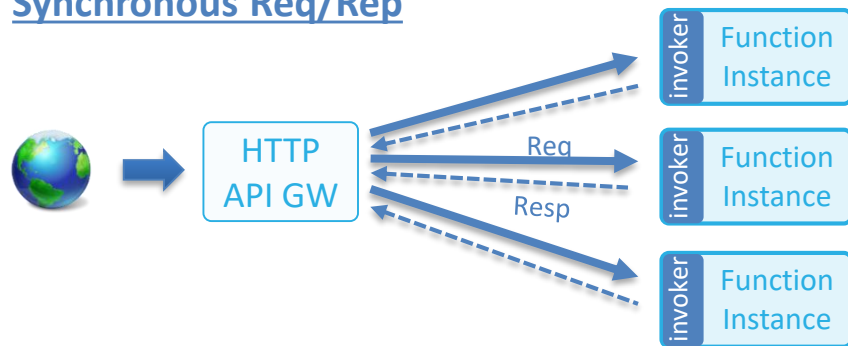
https://github.com/v3io/http_blaster

Native
Prometheus
Integration

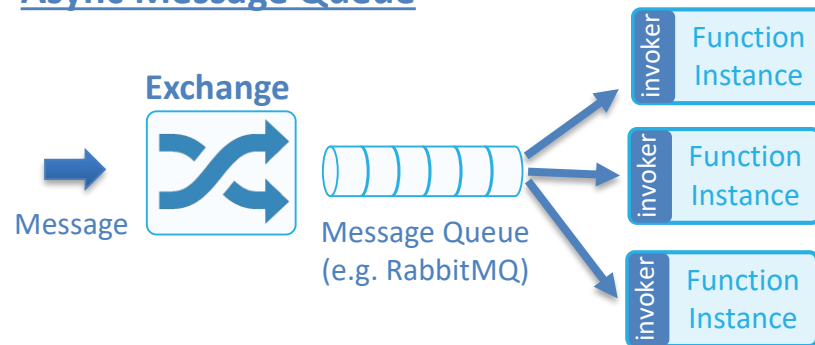


Nuclio Invocation Modes

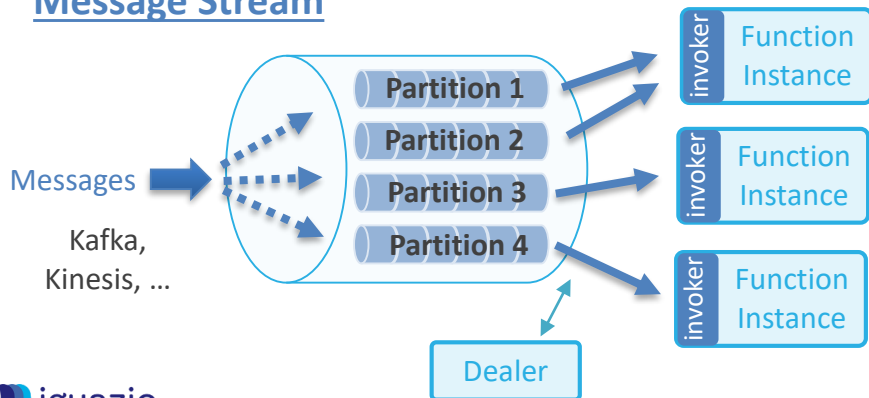
Synchronous Req/Rep



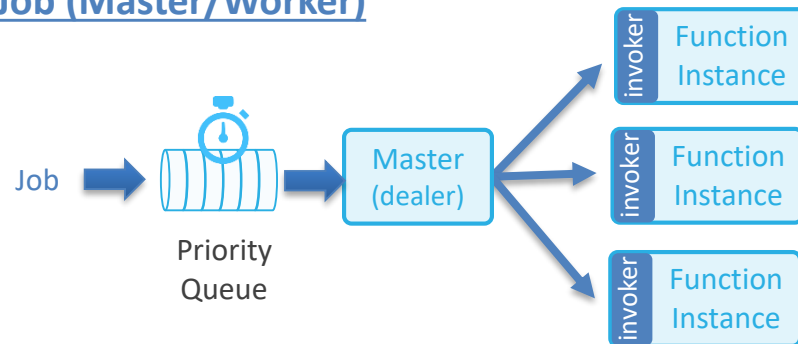
Async Message Queue



Message Stream

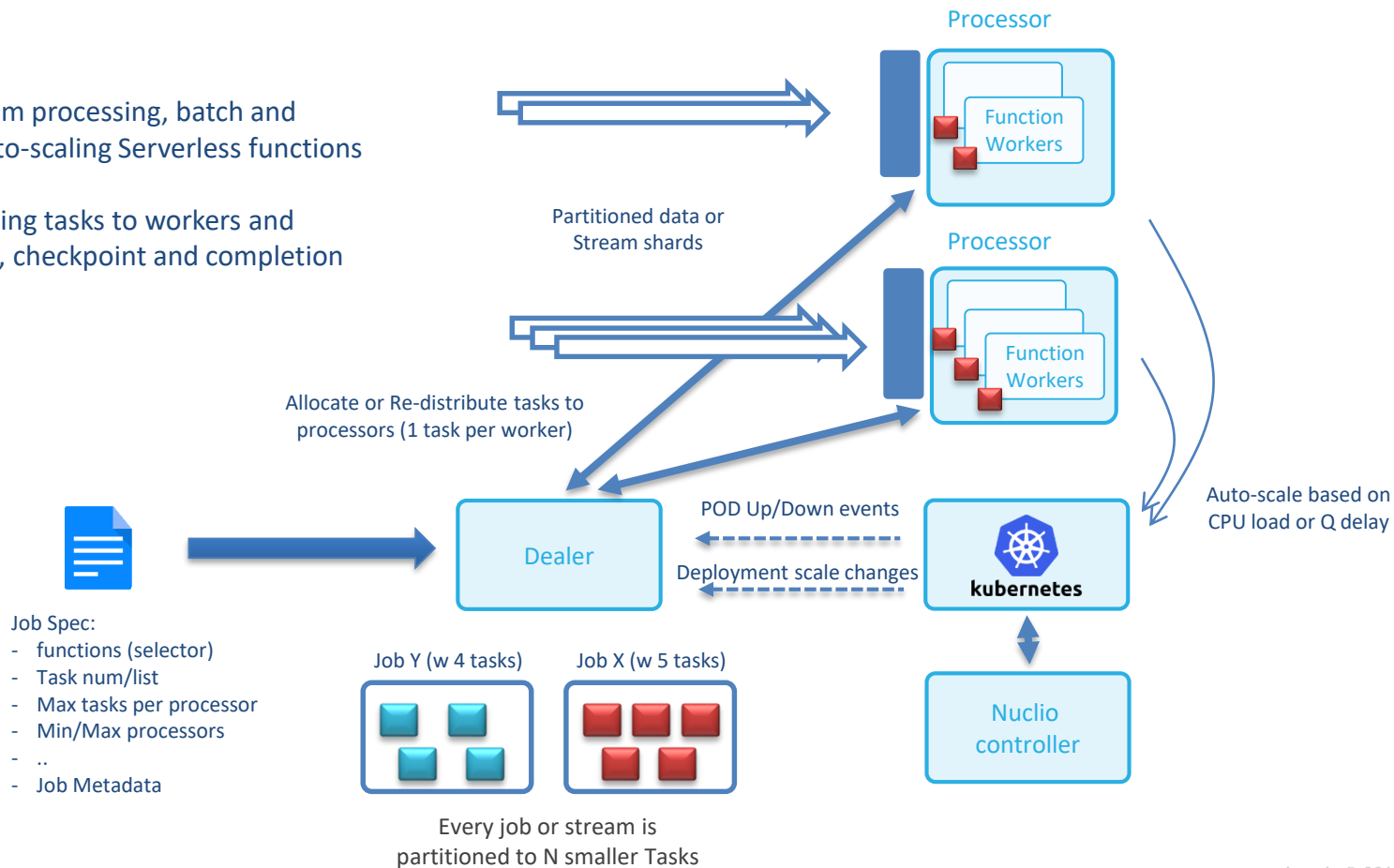


Job (Master/Worker)

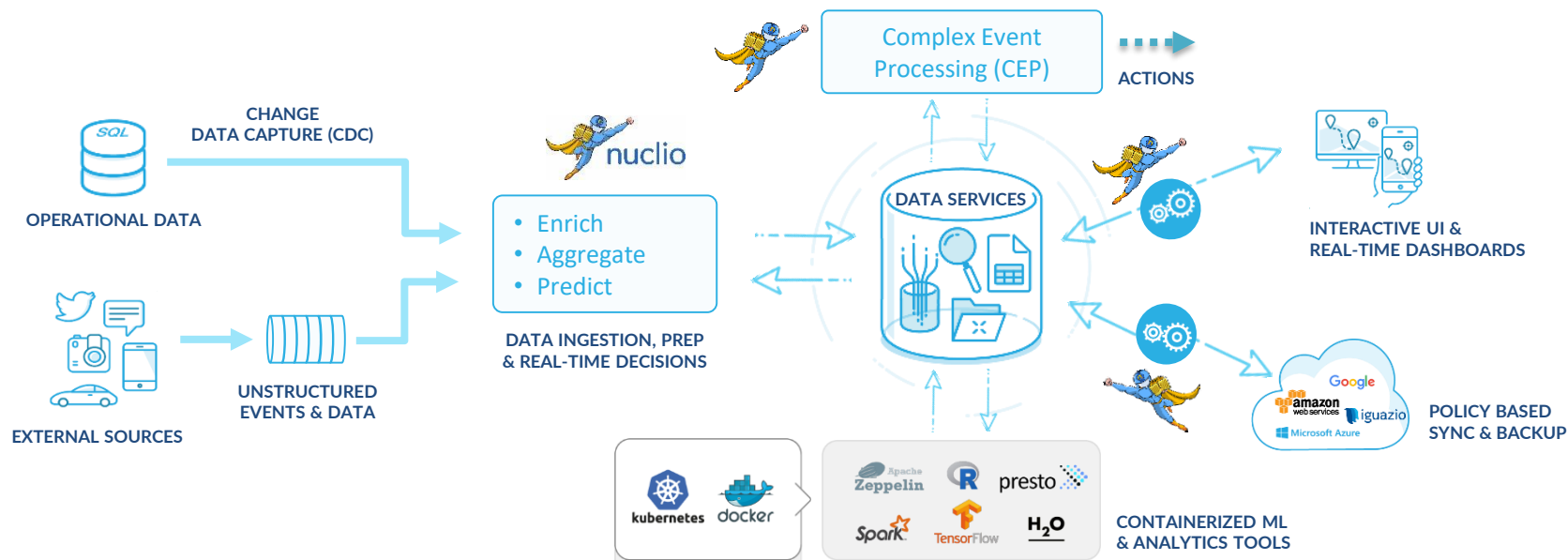


Nuclio Dealer

- 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



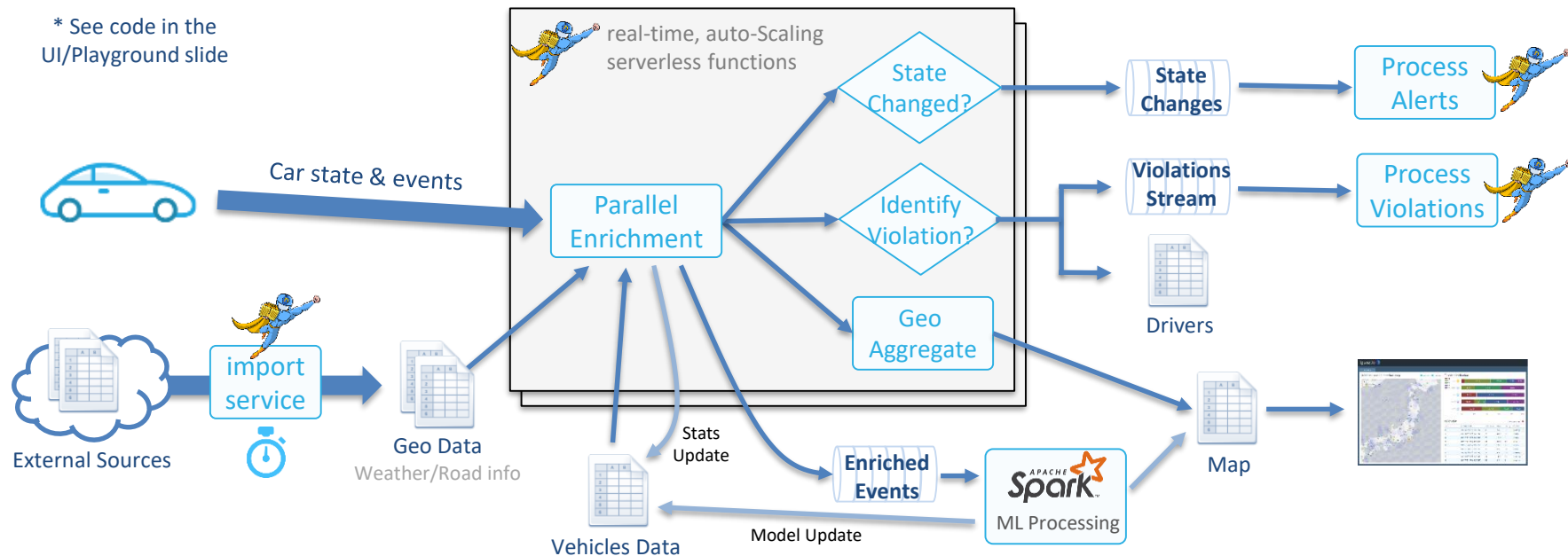
nuclo Features and Performance Make Serverless Broadly Applicable



Higher-Productivity | Faster insights | No Infrastructure Hassle | Lower TCO

Real Example: Event-Driven Analytics for Connected Cars

* See code in the
UI/Playground slide



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

nuclio

Function Spec

```

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

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

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

Nuclio Common Event Model

```

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

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{})

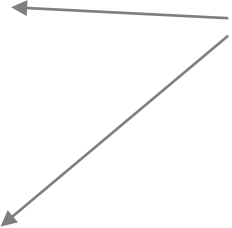
    // emit a structured log entry. example:
    //
    // l.InfoWith("The message",
    //     "first-key", "first-value",
    //     "second-key", 2)
    //
    ErrorWith(format interface{}, vars ...interface{})
    WarnWith(format interface{}, vars ...interface{})
    InfoWith(format interface{}, vars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

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

Nuclio Playground (run as isolated k8s deployment)


nuclio

Deploy

```

34
35 func CarEvent(context "nuclio.Context, event nuclio.Event) (interface{}, error) {
36     cont := context.DataBinding["db0"]
37
38     record := carEvent{
39         err := json.Unmarshal(event.GetBody(), &record)
40         if err != nil {
41             return nil, fmt.Errorf("Failed to Unmarshal json")
42         }
43         geoHash := GetGeoHash(record.Lon, record.Lat)
44
45         dc := dataframe.NewDataContext(context.Logger)
46
47         // Async (parallel) loading of cars and road info data
48         cars := dc.Read.FromTable(cont, "cars").Keys(record.VIN).LoadAsync()
49         roadMap := dc.Read.FromTable(cont, "roadinfo").Keys(geoHash).Select("weather", "speedlimit", "trafficCondition").LoadAsync()
50         car := cars.Next()
51         road := roadMap.Next()
52
53         if cars.Error() != nil || car==nil {
54             return nil, fmt.Errorf("Car %s not found", record.VIN)
55         }
56
57         // Sync (blocking) read of driver data, single row get by key (DriverID)
58         driver := dc.Read.FromTable(cont, "driver").Keys(car["DriverID"]).GetRow()
59
60         // Write Car, Road, and Driver data to the log at Info level
61         context.Logger.InfofWith("Got data", "car", car, "road", road, "driver", driver)
62
63         // Create an enriched data record to be streamed into various queues
64         enriched := enrichedEvent{carEvent:record, GeoHash:geoHash}
65         car.Scan("DriverID,state,lastGeoHash",&enriched.DriverID, &enriched.State, &enriched.LastGeoHash)
66         road.Scan("weather,trafficCondition",&enriched.Weather, &enriched.TrafficCondition)
67         enrichedMsg, err := json.Marshal(enriched)
68
69         // Write enriched data into stream (to be used by Spark Streaming for Machine Learning)
70         dc.Write.ToStream(cont, "enriched-stream").Records(enrichedMsg).SaveAsync()
71
72         // If car changed location update the geo map counters ( -1 in old location, +1 in new location)
73         if enriched.LastGeoHash != geoHash {

```

Configure

Invoke

POST
Send

Content type: JSON
Log level: Info

```
{
  "VIN": "1",
  "Speed": 19,
  "State": "ok",
  "Lat": "40.7513890",
  "Lon": "-73.9930560"
}
```

Log

Clear log

```
> Body:
{
  "VIN": "1",
  "Lon": "-73.9930560",
  "Lat": "40.7513890",
  "Speed": 19,
  "State": "ok",

```

CLI (run command example)

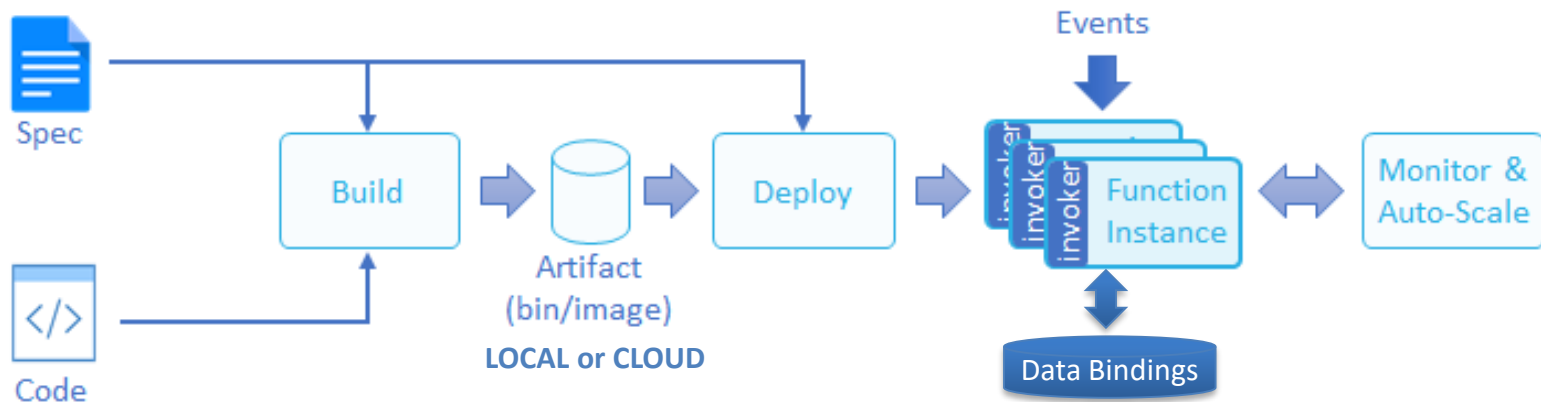
```
$ nuctl run --help
Build, deploy and run a function
```

```
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
```

Enabling Simple and Continuous Dev and Ops (CI/CD)



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

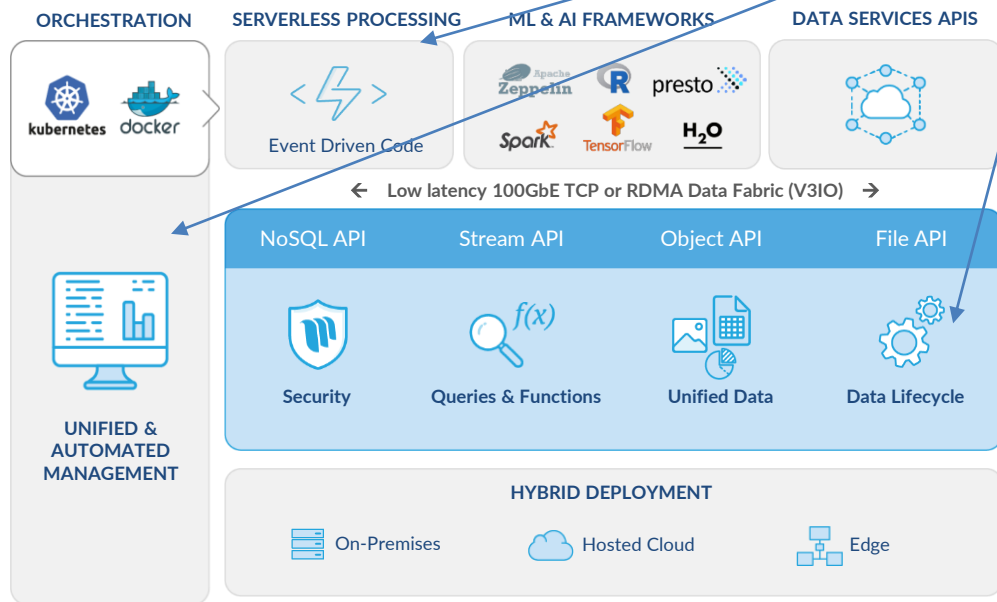
One Click to test, deploy, upgrade or rollback code
Runs ANYWHERE, Self-healing and Auto-Scaling





nuclo

<https://github.com/nuclo/nuclo>



- **Used in iguazio's 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