The Fn Project

Open Source Serverless Computing

“Go serverless....” Chicago Gophers Meetup

July 17, 2018

Dan Anderson, JDK Technologies

Who am I?

- Dan Anderson, Product Manager, JDK Technologies
- Checked out Go pre v1.0 to improve coding skills & acumen
- Post v1.0 used go for
  - Backend web servers for personal projects & POCs
  - General programming for scripts and utilities
  - Frequent use of Gin and mgo packages
- My Reasons to use Go
  - Static Typing/Compiled Language
  - Procedural structure
  - “Lightweight object orientation” through types and methods

JDK Technologies Story

JDK

- Help our client implement digital innovation to stay relevant, gain a competitive advantage, and build long term growth

- **True Systems** Approach
  - Combine Systems Thinking with Agile Product Development
  - Deliver holistic products rather than projects and code
  - Integrate into existing ecosystems and business processes

Objectives

- Brief, high level introduction to the Fn Project
- Get a flavor of the project and (hopefully) pique interest
- Highlight a Go based project

What is Serverless?

- **Serverless** is an abstraction of infrastructure and its operations including provisioning, scaling, patching, etc.
- **Serverless architecture** is when an app is built entirely on serverless components (compute, storage, networking)
- **Faas** is the compute component in a serverless architecture

Chicago Gophers

fn

Functions-as-a-Service

- **Functions** are small bits of code that do one thing well and are easy to understand and maintain
- **As a service** means no complicated plumbing, the system takes care of provisioning, scaling, patching, maintaining, etc. Each function scales independently.

Introducing the Fn Project

- Open-source serverless compute platform
- Can be deployed to any cloud and on-premise
- Simple, elegant, and extensible by design
- Containers are primitives
- Active w/ 2500+ commits across 50+ contributors
- Independently governed with plans for foundation
- Independent yet vendor backed
- Strong enterprise focus (security, scalability, observability, etc.)

Why the Fn Project

- Open Source
- Multi Cloud
- Container Native
- Orchestrator Agnostic

See: Medium.com - 8 Reasons why we built the Fn Project by Chad Arimura *link*

Why Did I Check Out the Fn Project?

- Method to start learning about FaaS concepts
- Impressed by Founders Experience at Iron.io
    - Iron.io: early Go adopter
    - Iron.io Founders Chad Arimura (CEO) and Travis Reeder (CTO) now drivers of the Fn Project
- Can develop locally and in the cloud
- Stand up common asynchronous tasks
    - Log events
    - Send messages and alerts
    - Simple file processing

An Fn Function

- Small chunk of code wrapped into a container image
- Gets input via STDIN and environment
- Produces output to STDOUT
- Logs to STDERR

The Fn server handles everything else, like the API gateway, piping things around, storing logs, etc.

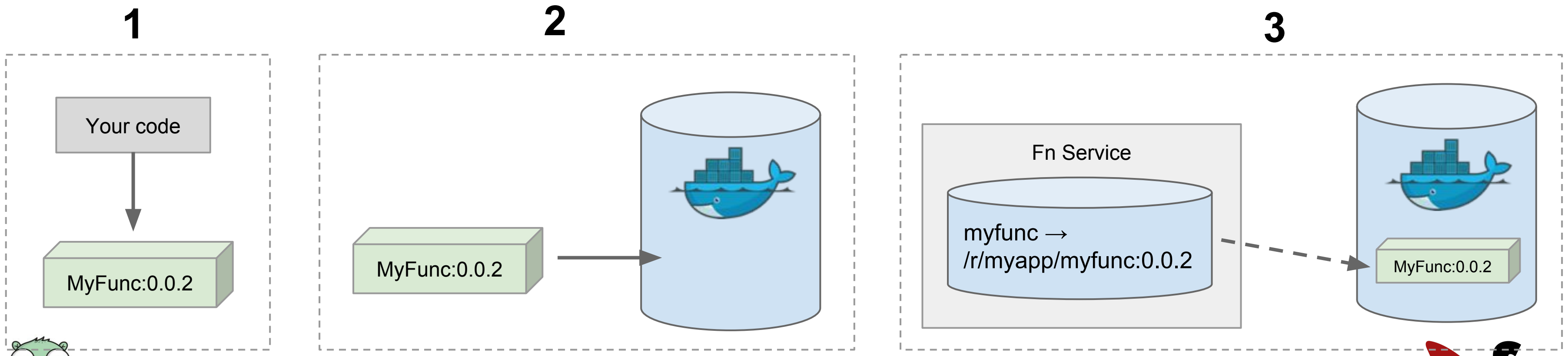
Fn CLI

- **fn init** --runtime go
- **fn run**
- **fn test**
- **fn deploy** --app myapp
- **fn call** myapp myfunc

→ http://localhost:8080/r/myapp/myfunc

fn deploy details

1. Builds container (multi-stage) + bumps version
2. Pushes container to registry
3. Creates/updates function route (servers lazy load images)

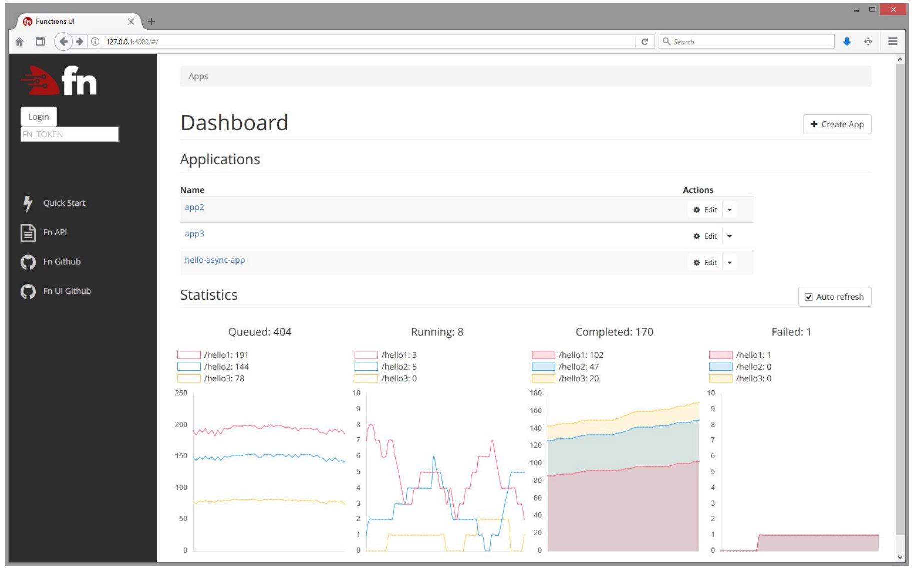


Function Development Kits (FDKs)

- Used to help with parsing input and writing output
- Familiar syntax for Lambda developers
- Simply write a `handler` function that adheres to the FDK’s interface and it will parse STDIN and provide the input data to your function and deal with writing the proper output format.
- Makes it a lot easier to write hot functions

Fn UI

```
docker run --rm -it --link fnserver:api -p 4000:4000 -e
"FN_API_URL=http://api:8080" fnproject/ui
```



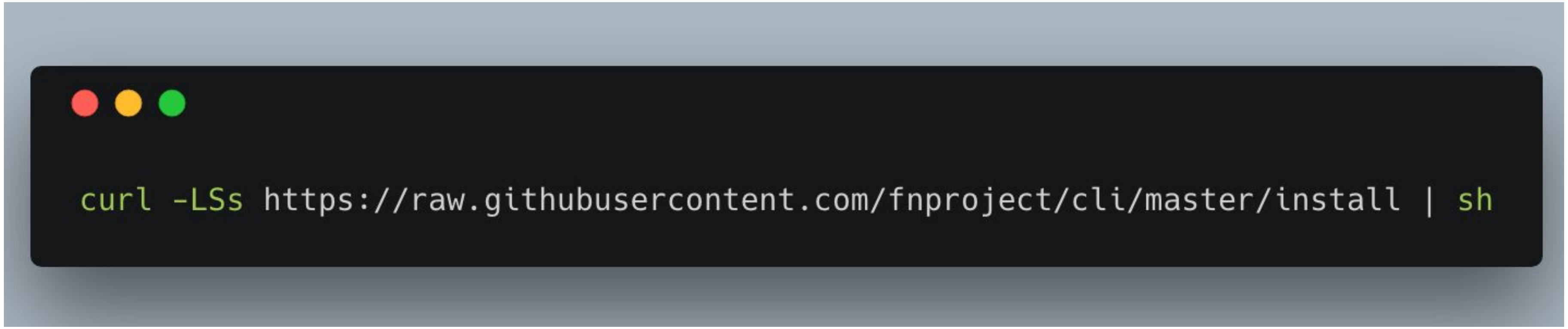
Name	Actions
app2	Edit
app3	Edit
hello-async-app	Edit

```
Queued: 404          Running: 8          Completed: 170        Failed: 1
/hello1: 191         /hello1: 3          /hello1: 102          /hello1: 1
/hello2: 144         /hello2: 5          /hello2: 47           /hello2: 0
/hello3: 78          /hello3: 0          /hello3: 20           /hello3: 0
```

Install Fn Server

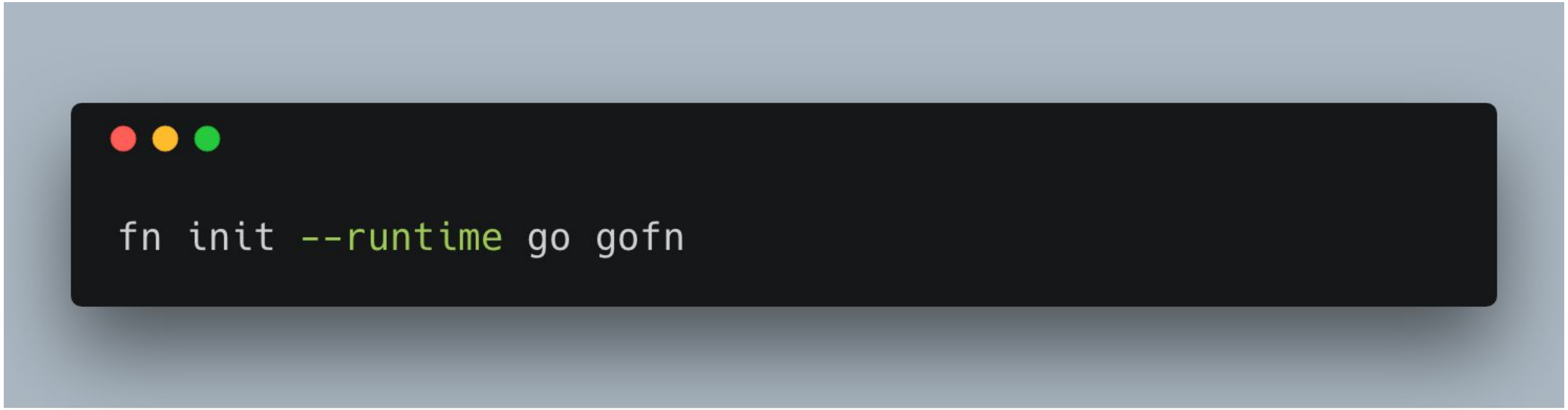
Prerequisites

- Linux or MacOS
- Docker v17.10 or higher
- Docker Hub account



```
curl -LSs https://raw.githubusercontent.com/fnproject/cli/master/install | sh
```

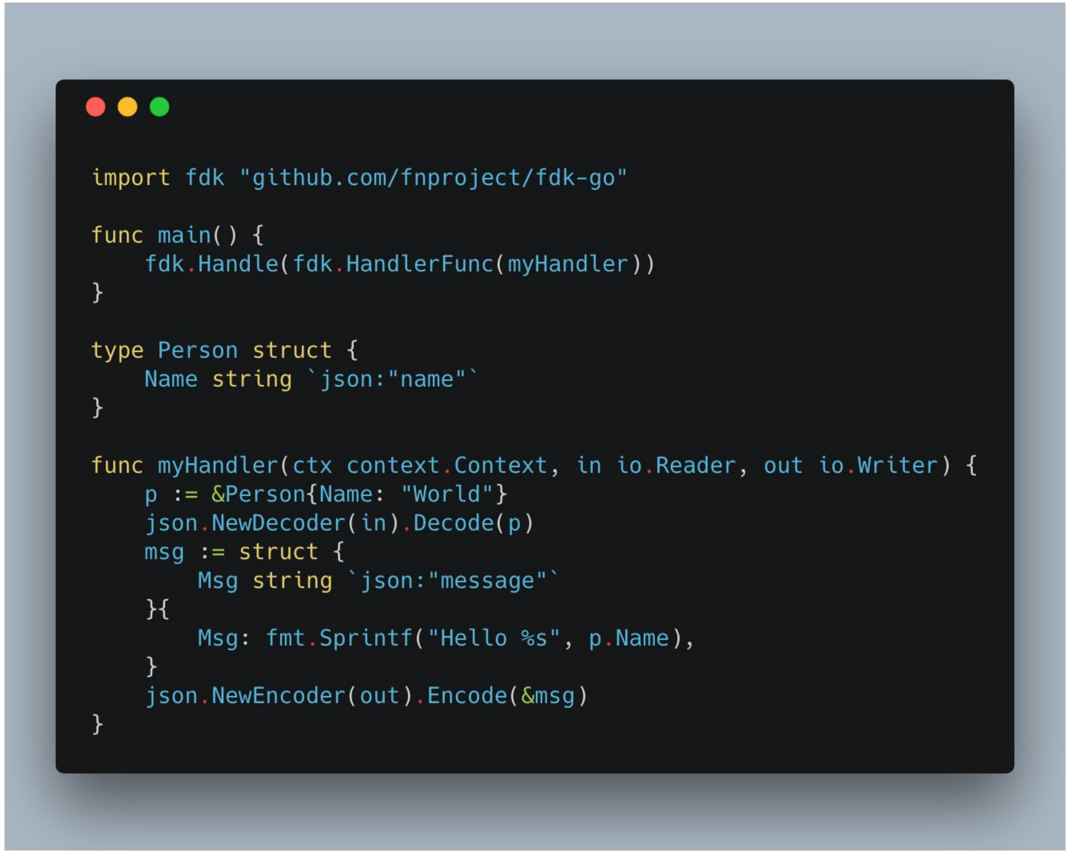
Initialize a “Hello World” Function



```
fn init --runtime go gofn
```

Chicago Gophers

Function Code



```go
import fdk "github.com/fnproject/fdk-go"

func main() {
    fdk.Handle(fdk.HandlerFunc(myHandler))
}

type Person struct {
    Name string `json:"name"`
}

func myHandler(ctx context.Context, in io.Reader, out io.Writer) {
    p := &Person{Name: "World"}
    json.NewDecoder(in).Decode(p)
    msg := struct {
        Msg string `json:"message"`
    }{
        Msg: fmt.Sprintf("Hello %s", p.Name),
    }
    json.NewEncoder(out).Encode(&msg)
}
```

Deeper Topics

- Fn Load Balancer
- Hot Functions
- Middleware
- Fn Flow
- Security

Thank you!

Dan Anderson
Product Manager, JDK Technologies
dan@jdktech.com
www.jdktech.com

- Github: github.com/fnproject/fn
- Slack: slack.fnproject.io
- Learn more: fnproject.io
- Today's Code: github.com/danoand/fn-meetup

Appendix

The Fn Project

Fn Server

FDK’s

Fn Flow

Debugging

- **fn calls list** myapp
- **fn calls get** myapp <call-id>
- **fn logs get** myapp <call-id>
- Metrics created using OpenTracing w/ initial collectors and extensions for Prometheus, ZipKin, and soon Jaeger