Botless: Building a Serverless Chatbot Framework
We are going to build the botless framework *together*, right now. We will optimize for **price**, and **flexibility** of extensions.
Collect Requirements
CRD stands for Custom Resource Definition.
Serverless

- No servers
- Elastic
- Portable
- Service-based
- Event-driven
- Pay only for usage
Standardize
Knative

Building blocks for serverless workloads on Kubernetes
Knative project

- Set of primitives (build, events, serving...)
- Solves for modern development patterns
- Implements learnings from Google, partners
- Ingredient or platform for Serverless

github.com/knative
Knative collaboration
Knative enabled

Products
- Serverless Containers on GCF
- GKE Serverless Add-on
- SAP Kyma
- Pivotal Function Service
- IBM Cloud Functions
- Red Hat Cloud Functions
- Pivotal riff
- OpenFaaS
- T-mobile Jazz

Primitives
- Build
- Serving
- Eventing
- ...

Platform
- Kubernetes

Google Cloud
General Architecture
Chat Server Protocol.

Outputs chat messages in to Chat Message Cloud Event.
CloudEvents

a specification for describing event data in a common way.
CloudEvents

```json
{
    "cloudEventsVersion": "0.1",
    "eventType": "com.example.someevent",
    "source": "http://example.com/mycontext",
    "eventID": "A234-1234-1234",
    "eventTime": "2018-04-05T17:31:00Z",
    "contentType": "text/xml",
    "data": "<much wow="xml="/"/>
}
```
apiVersion: sources.botless.dev/v1alpha1
kind: ChatSource
metadata:
  name: example-chat-source
spec:
  server:
    host: http://chatserver.example.com
  auth:
    username: botless
    password:
    ref:
      name: botless-chatserver
      key: botlessCredentials
  sink:
    apiVersion:
      eventing.knative.dev/v1alpha1
    kind: Channel
    name: msg
Knative, principled eventing objects

Eventing constructs:

- **sources** (producer)
- **Channel** (delivery)
- **Subscription** (linkage)
- **callable** (custom functions)
apiVersion: eventing.knative.dev/v1alpha1
kind: Channel
metadata:
  name: k8ssource
spec:
  provisioner:
    name: default
Knative, subscriptions

apiVersion: eventing.knative.dev/v1alpha1
type: Subscription
metadata:
  name: sub-example
  namespace: default
spec:
  channel:
    apiVersion: eventing.knative.dev/v1alpha1
    kind: Channel
    name: upstream-channel
  subscriber:
    ref:
      apiVersion: serving.knative.dev/v1alpha1
      kind: Service
      name: caller-example
  reply:
    channel:
      apiVersion: eventing.knative.dev/v1alpha1
      kind: Channel
      name: downstream-channel
Knative, fanout
Knative, callable

passthrough
Knative, callable
Knative, callable

filter
Applied Eventing
{  
  "text": "/uptime",
  "user": "n3wscott",
  "room": "default"
}

{  
  "cloudEventsVersion": "0.1",
  "eventType": "dev.botless.chat",
  "source": "http://botless/source",
  "eventID": "B134-5678-3251",
  "correlationID": "1ACB-2123-JKE3",
  "eventTime": "2018-04-05T17:28:00Z",
  "contentType": "application/json",
  "data": {  
    "message": "/uptime",
    "author": "n3wscott"
  }
}
CloudEvents, correlating events

```json
{
  "cloudEventsVersion": "0.1",
  "eventType": "dev.botless.chat",
  "source": "http://botless/source",
  "eventID": "B134-5678-3251",
  "correlationID": "1ACB-2123-JKE3",
  "eventTime": "2018-04-05T17:28:00Z",
  "contentType": "application/json",
  "data": "...
}
```

```json
{
  "cloudEventsVersion": "0.1",
  "eventType": "dev.botless.command",
  "source": "http://botless/parser",
  "eventID": "A234-1234-1234",
  "correlationID": "1ACB-2123-JKE3",
  "eventTime": "2018-04-05T17:31:00Z",
  "contentType": "application/json",
  "data": "...
}
```
Knative, principled serving objects

Knative defines primitives with clear separation of concerns

**Configuration**
Current/desired state for your application
Code & configuration (separated, ala 12 factor)

**Revision**
Point in time snapshots for your code and configuration

**Route**
Maps traffic to a revisions
Supports fractional, named routing
Encapsulate Complexity
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: chatgateways.sources.botless.dev
spec:
  group: sources.botless.dev
  names:
    categories:
    - sources
    kind: ChatGateway
    plural: chatgateways
  scope: Namespaced
  version: v1alpha1

Chat Gateway
apiVersion: sources.botless.dev/v1alpha1
kind: ChatGateway
metadata:
  name: example-gateway
spec:
  server:
    host: http://chatserver.example.com
  auth: ...
sink:
  apiVersion: eventing.knative.dev/v1alpha1
  kind: Channel
  name: msg
response:
  apiVersion: eventing.knative.dev/v1alpha1
  kind: Channel
  name: resp
Chat Gateway

Chat Server

Receive Adaptor (Source)

Transmit Adaptor
Chat Gateway

Parser

Router

def

resp

msg

cmd

</>

//foo

c=f

</>
Expand
/learn demo https://github.com/botless/demo-command/cmd/demo

Learning...

/demo Ready!

/demo

Hello KubeCon China!
Knative, principled build objects

TODO
Botless, new custom framework

Leverages Knative for:
● Serverless functions
● Eventing Connections
● Container Building

Leverages Kubernetes for:
● Framework Portability
● Custom Extension (CRDs)
Takeaways, consider the following

Customize Kubernetes
● Extend k8s to encapsulate your software
● Careful to design in composability between resources

Knative is an example of a framework built on top of k8s.

Botless is a purpose built custom framework as an example, you can do this too!
Thank you