

# When the Logs Just Don't Cut It: Root-Causing Incidents Without Re-Deploying Prod

*Phillip Kuznetsov*



**KubeCon**



**CloudNativeCon**

North America 2022

BUILDING FOR THE ROAD AHEAD

**DETROIT 2022**

**October 24-28, 2021**

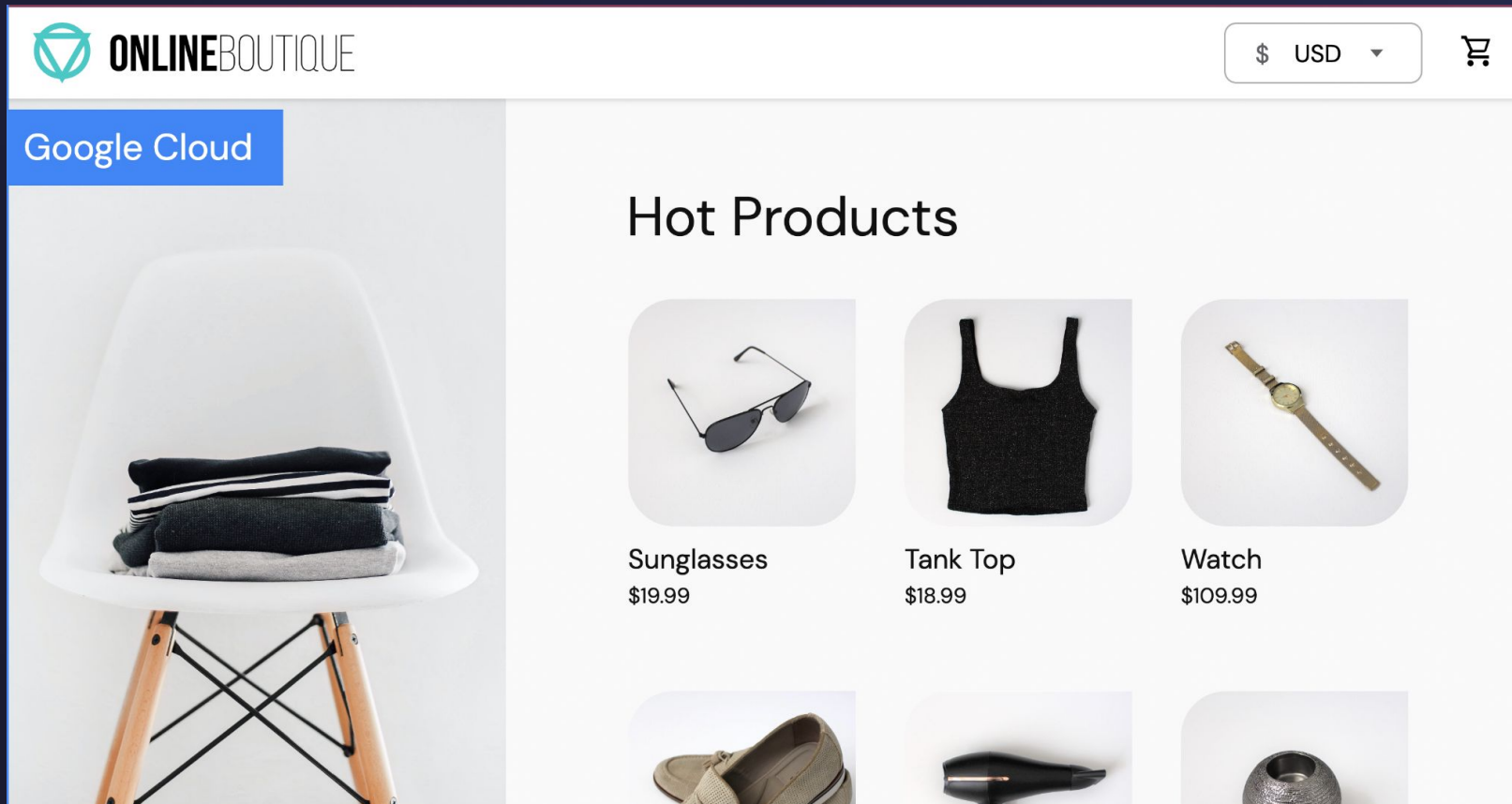


**Phillip Kuznetsov**

Principal Software Engineer

*New Relic*

# The Situation



<https://github.com/GoogleCloudPlatform/microservices-demo>

# Incident

Our frontend service is  
panicking



# New Product Item



**ONLINE**BOUTIQUE

**Sticker**

**\$0.99**

# The Search

# Let's check the logs

```
Logs(px-online-boutique/frontend-75cc5bb58c-6tlct:server)[1m]
Autoscroll:Off FullScreen:Off Timestamps:Off Wrap:Off

created by net/http.(*Server).Serve
    /usr/local/go/src/net/http/server.go:3102 +0x4db
{"http.req.id":"776cc819-5bbf-4084-8ce8-b16fb2ea1cee","http.req.method":"GET","http.req.path":"/cart","message":"req
{"http.req.id":"776cc819-5bbf-4084-8ce8-b16fb2ea1cee","http.req.method":"GET","http.req.path":"/cart","message":"vie
{"http.req.id":"776cc819-5bbf-4084-8ce8-b16fb2ea1cee","http.req.method":"GET","http.req.path":"/cart","http.resp.by
2022/10/27 14:46:48 http: panic serving 10.48.0.81:65027: one of the specified money values is invalid
goroutine 24747 [running]:
net/http.(*conn).serve.func1()
    /usr/local/go/src/net/http/server.go:1850 +0xbf
panic({0xa3cee0, 0xc000226690})
    /usr/local/go/src/runtime/panic.go:890 +0x262
github.com/GoogleCloudPlatform/microservices-demo/src/frontend/money.Must(...)
    /src/money/money.go:85
main.(*frontendServer).viewCartHandler(0xc19d20?, {0xc19148?, 0xc0005ca3a0}, 0xc00011b900)
    /src/handlers.go:297 +0x162e
net/http.HandlerFunc.ServeHTTP(0xc00011b800?, {0xc19148?, 0xc0005ca3a0?}, 0x0?)
    /usr/local/go/src/net/http/server.go:2109 +0x2f
```

panic serving 10.48.0.81:65027: one of the specified money values is invalid

# We keep seeing this error:

```
ErrInvalidValue      = errors.New("one of the specified money values is invalid")
```



... which is returned here

```
// both).  
func Sum(l, r pb.Money) (pb.Money, error) {  
    if !IsValid(l) || !IsValid(r) {  
        return pb.Money{}, ErrInvalidValue
```

# The error is not helpful

```
errors.New("one of the specified money values is invalid")
```

What value is invalid??

# What's inside the arg?

```
// IsValid checks if specified value has a valid units/nanos signs and ranges.  
func IsValid(m pb.Money) bool {  
    return signMatches(m) && validNanos(m.GetNanos())  
}
```

# pb.Money

```
type Money struct {  
    Units          int64  
    Nanos          int32  
    CurrencyCode   string  
}
```

# What's inside the arg?

```
// IsValid checks if specified value has a valid unit  
func IsValid(m pb.Money) bool {  
    return signMatches(m) && validNanos(m.GetNanos())  
}
```

# Can we just add a new log?

```
// IsValid checks if specified value has a valid unit
func IsValid(m pb.Money) bool {
    log.Infof("isValid: %v", m)
    return signMatches(m) && validNanos(m.GetNanos())
}
```

**... if only it were so easy**

Reproducing a bug outside of prod might not work

Deploying to Prod takes too long

Deploying to Prod is risky

Deploying to Prod is out of compliance

# What if there's another way?





# bpftrace

**bpftrace**

Public

High-level tracing language for Linux  
eBPF



C++



5.9k

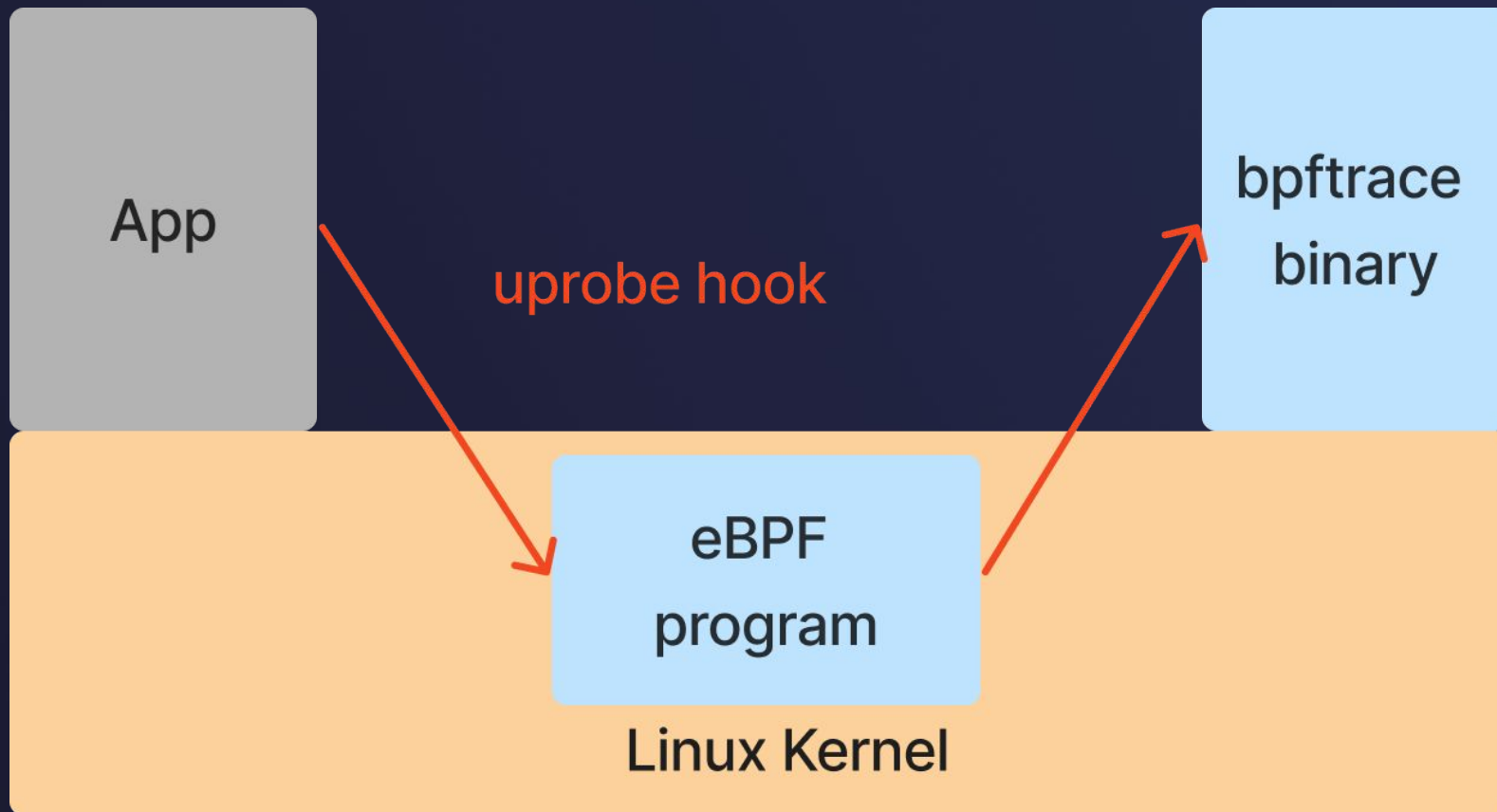


925

<https://github.com/iovisor/bpftrace>

# Sandboxed programs running in the Kernel



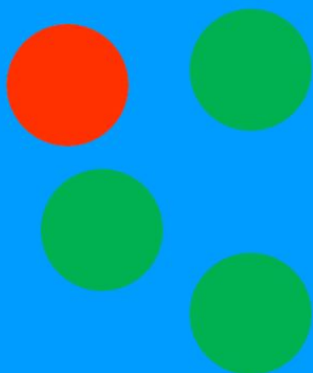


1. Pause execution
2. Run bpftrace/eBPF program
3. Resume execution

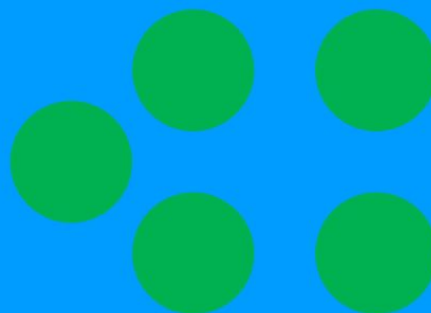
```
uprobe:"github.com/[...]/frontend/money.IsValid" {  
    $units = reg("ax");  
    $nanos = (int32) reg("bx");  
    printf("time_:%lld, units:%d, nanos:%d",  
          nsecs, $units, (int64)$nanos);  
}
```

## Cluster

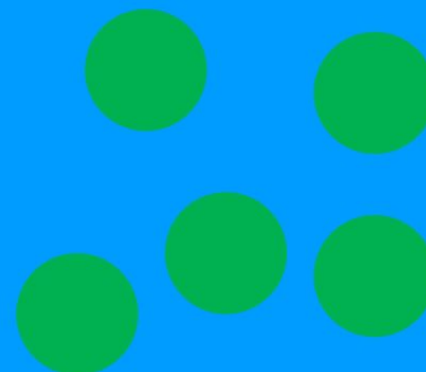
Node



Node



Node





# PIXIE



**CLOUD NATIVE**  
**SANDBOX**

<https://github.com/pixie-io/pixie>

```
name = "frontendIsValid"
pxtrace.UpsertTracepoint(
    name,
    name,
    program,
    target=pxtrace.LabelSelectorProcess(
        {'app': 'frontend'},
        namespace='px-online-boutique',
        container_name='server',
    ), ttl='5m'
)
```

<https://github.com/pixie-io/pixie>

bpfttrace + Pixie =  
print() statements in prod

[docs.px.dev/tutorials/custom-data/distributed-bpfttrace-deployment/](https://docs.px.dev/tutorials/custom-data/distributed-bpfttrace-deployment/)



# Demo



```
1  import px
2  import pxtrace
3
4  program = """
5  uprobe:"github.com/GoogleCloudPlatform/microservices-demo/src/frontend/money.IsValid" {
6      $units = reg("ax");
7      $nanos = (int32) reg("bx");
8      printf("time_:%lld, units:%d, nanos:%d, stack:%s", nsecs, $units, (int64)$nanos, ustack());
9  }
10 """
11 name = "frontendIsValid"
12 pxtrace.UpsertTracepoint(
13     name,
14     name,
15     program,
16     target=pxtrace.LabelSelectorProcess(
17         {'app': 'frontend'},
18         namespace='px-online-boutique',
19         container_name='server',
20     ), ttl='5m'
21 )
22
23 df = px.DataFrame(name, start_time='-5m')
24 px.display(df)
```

# Takeaways

Inserted logs into a running  
Kubernetes Pod

Used logs to determine the  
root-cause of a tricky incident



# Learn about the tools

bpftrace tools:











<https://github.com/iovisor/bpftrace#tools>

(38 at current moment)

Pixie - K8s Native Observability:

<https://github.com/pixie-io/pixie/>

# Cool eBPF Projects

Domain	Projects
Networking	 cilium 
Observability	 bpfttrace   PyroScope   Inspektor Gadget
Security	 Falco  Tracee  KubeArmor



# Great Links

1. bpftrace guide

[github.com/iovisor/bpftrace/blob/master/docs/reference\\_guide.md](https://github.com/iovisor/bpftrace/blob/master/docs/reference_guide.md)

2. What is eBPF? [ebpf.io/what-is-ebpf](https://ebpf.io/what-is-ebpf)

3. go-bpf-gen: [github.com/stevenjohnstone/go-bpf-gen](https://github.com/stevenjohnstone/go-bpf-gen)

4. Tracing SSL/TLS connections with eBPF:

[blog.px.dev/ebpf-openssl-tracing/](https://blog.px.dev/ebpf-openssl-tracing/)

5. Challenges of BPF Tracing Go:

[blog.0x74696d.com/posts/challenges-of-bpf-tracing-go/](https://blog.0x74696d.com/posts/challenges-of-bpf-tracing-go/)



Please scan the QR Code above to  
leave feedback on this session