Cloud Native Observability

With Prometheus, ElasticStack and Jaeger

whoami

- Alexander Knipping, Systems Engineer @ noris network AG
  - Kubernetes, Prometheus
  - Cloud Native Adoption
- ❤️ Open Source
  - https://github.com/obitech
- Co-Host of CNCF Nürnberg Meetup
  - https://www.meetup.com/Kubernetes-Nurnberg/
  - https://github.com/k8s-nue-meetup/talks

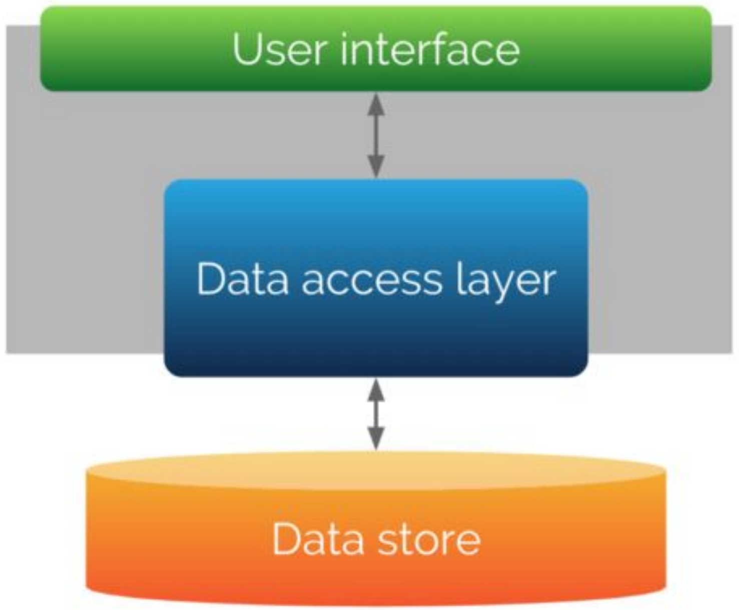
Agenda

1. Problem Space
    a. Why are we talking about this?
    b. What is Observability?
2. Tools & Demo
    a. Demo Microservice Application
    b. Prometheus, ElasticStack, Jaeger
3. Q&A, Feedback

noris network

Why are we talking about this?

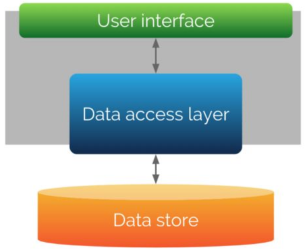
The Traditional Architecture



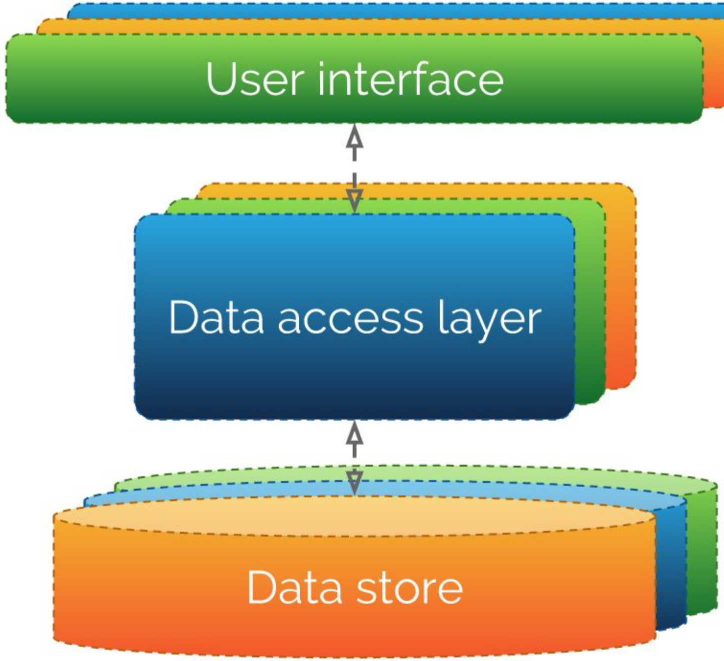
blog.heptio.com/

The Traditional Architecture

- Easy to develop, deploy, test, monitor
- Scaling becomes inefficient
  - Up vs. out
  - Only scales in regards to transactions
  - Only parts need to scale
- Code base grows
  - Introduces tight coupling, side effects
  - Onboarding new developers becomes harder
  - Deployments are tricky
- Big machines are expensive

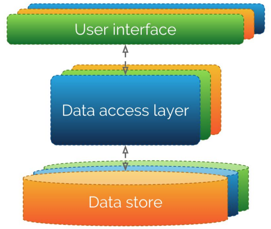


Microservices & Distributed Systems



blog.heptio.com/

Microservices & Distributed Systems



- Small services
  - Easy to understand & reason about
  - Faster testing, building, deploying
  - Loosely coupled
- Higher isolation
  - Single degraded service is unlikely to bring down whole system
  - Independent deployments
- DevOps required
- Increased infrastructure complexity
  - Distributed System
  - Deployment complexity
  - Losing visibility/**observability**

What is Observability?

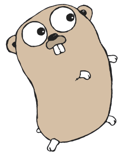
- What is going on in my system?
- Monitoring
  - Resource metrics (CPU, RAM, Disk, Net, ...)
  - Latency distributions
  - Error rates
  - Application & business metrics
- Logging
  - Syslog streams/files
  - Application Logs
- Tracing
  - Which service is talking with whom and for how long?
  - Communication graphing

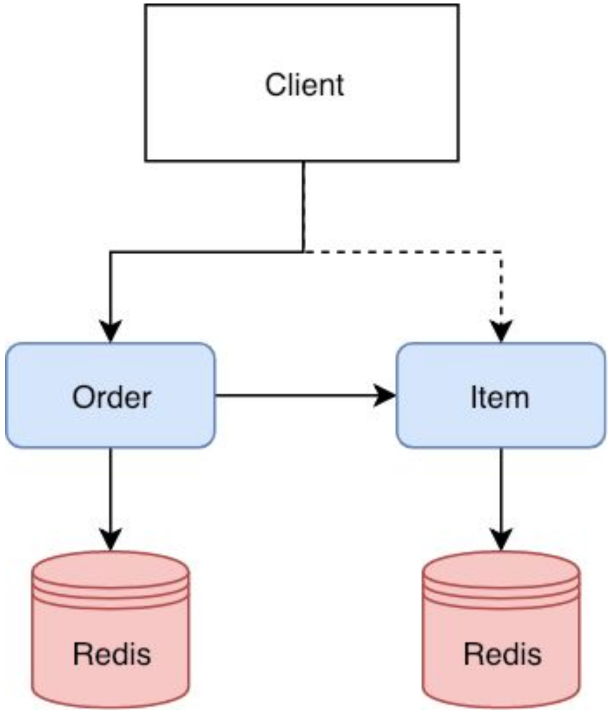
Tools & Demo

Demo Microservice Application

https://github.com/obitech/micro-obs

- Two services, REST APIs
- User → Item
- User → Order → Item → Order
- Instrumented directly (no sidecar)
- Written in Go





noris network

Cloud Native Observability

… using Open Source tools:

Monitoring with Prometheus

Resource utilization, error rate, latency, etc.

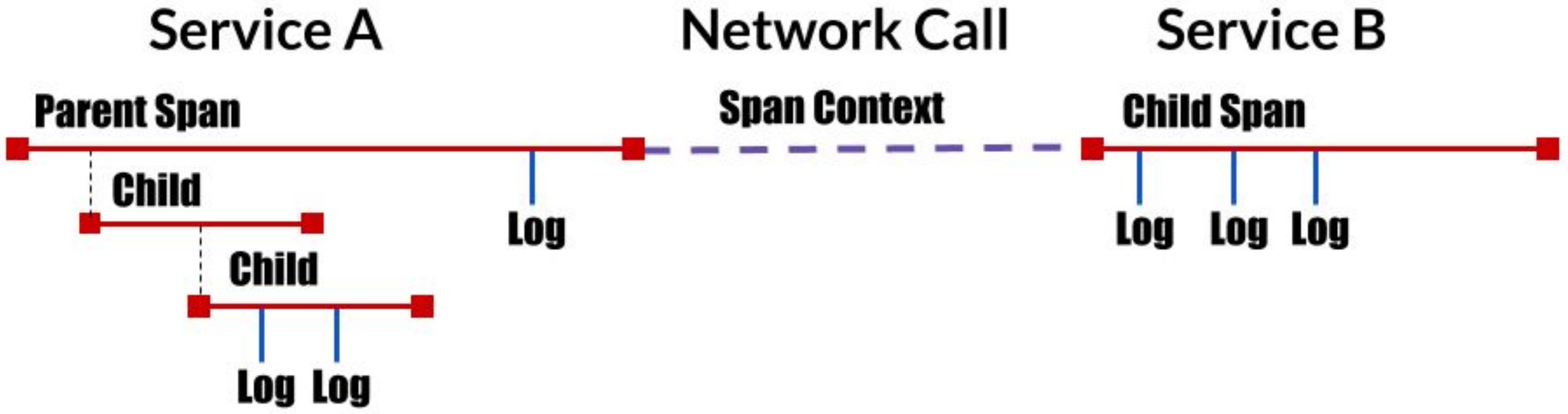
Centralised Logging with ElasticStack

Gather & query service logs

Distributed Tracing with OpenTracing & Jaeger

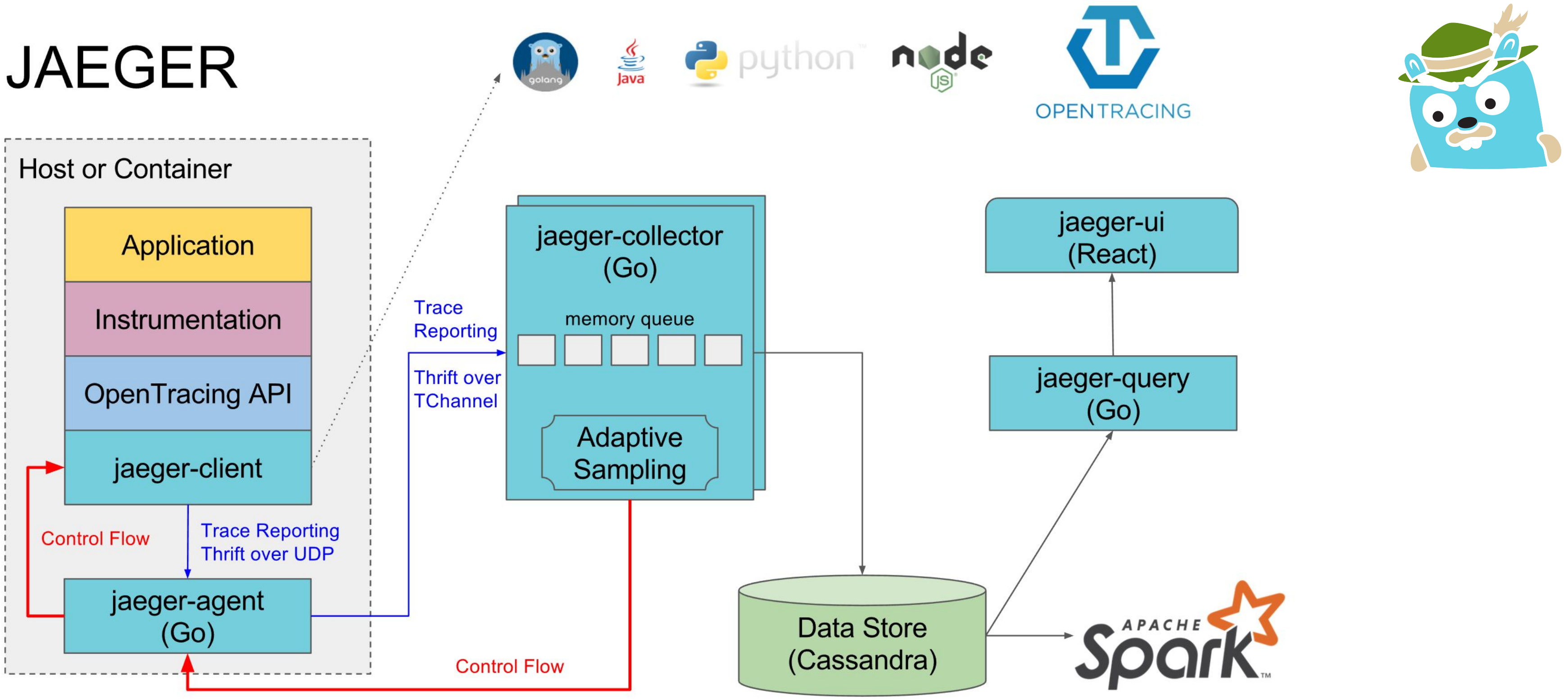
Visualize service communication

OpenTracing / OpenTelemetry



https://opentracing.io/docs/overview/

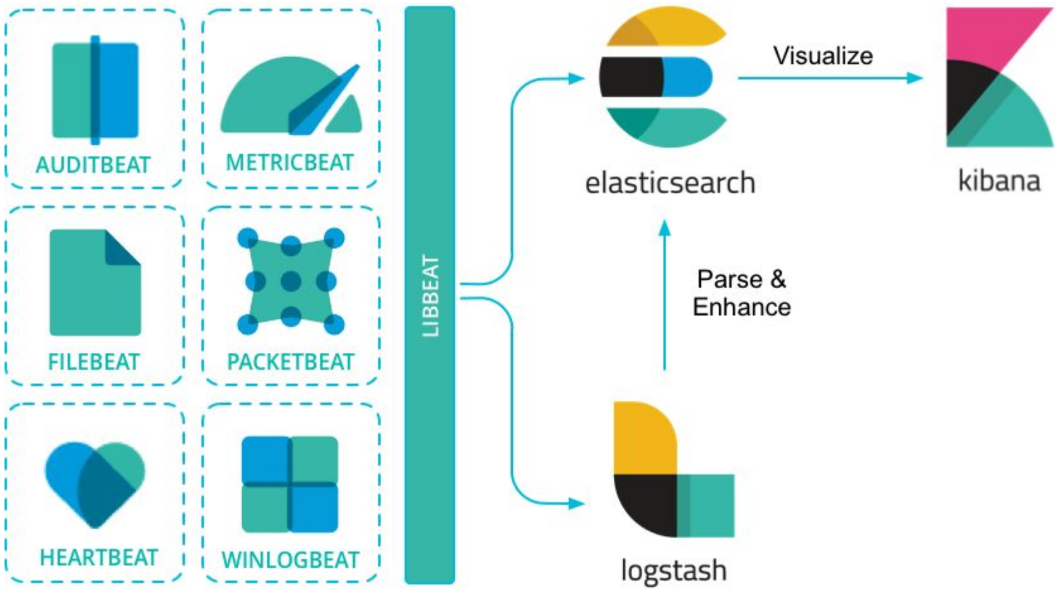
JAEGER



https://www.jaegertracing.io/docs/1.8/architecture/

Demo: Jaeger

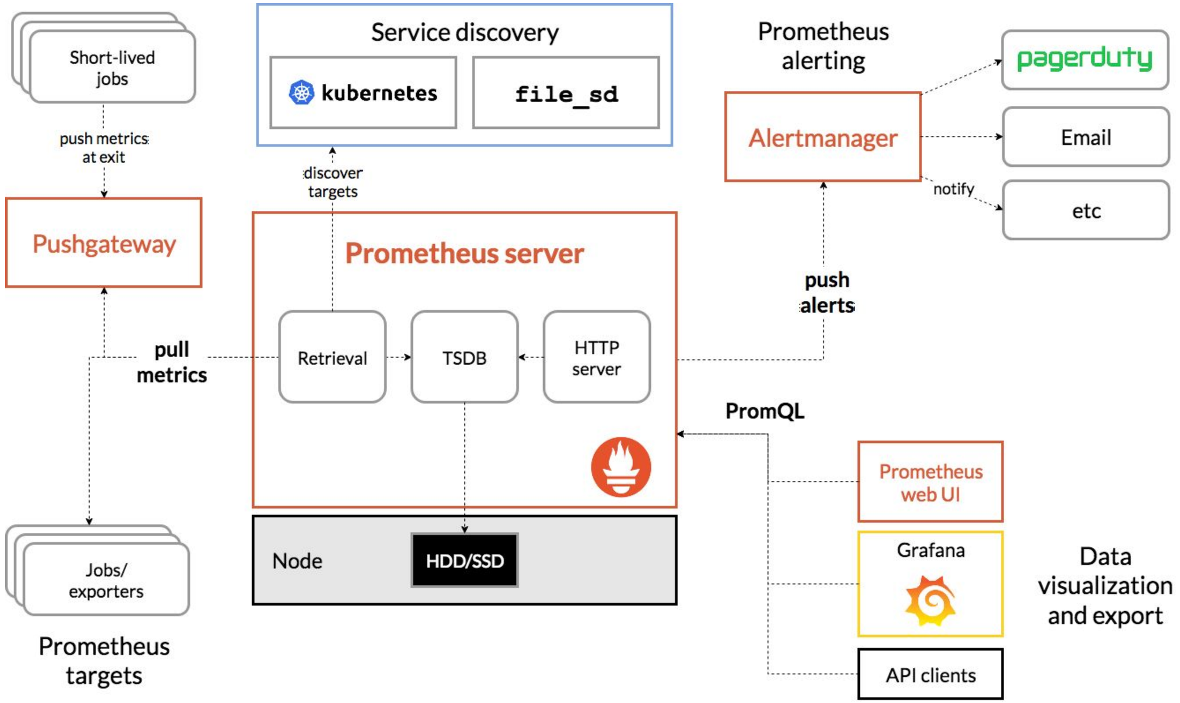
ElasticStack



https://www.elastic.co/guide/en/beats/libbeat/current/beats-reference.html

Demo: ElasticStack

Prometheus



https://prometheus.io/docs/introduction/overview/

Demo: Prometheus

Demo: Show some code?

Final thoughts

- Follow best practices
  - Structured logs
  - Log request IDs, not trace IDs
  - Monitoring...
    - Resources: USE method
    - Endpoints: RED method
- Simple instrumentation, no overhead for application
- But: complex observability stack
- Logstash & ElasticSearch = memory hungry (~50%, 1GB)

Thank you!