








ARAF KARSH HAMID

Co-Founder / CTO

MetaMagic Global Inc., NJ, USA

 @arafkarsh

  arafkarsh
 

Building Cloud Native Apps

Microservice Architecture Series

Service Mesh / Istio

Zipkin / Prometheus / Grafana / Kiali

Monitoring / Observability

1

Monitoring
Observability

Zipkin
Prometheus
Grafana / Kiali

3

2

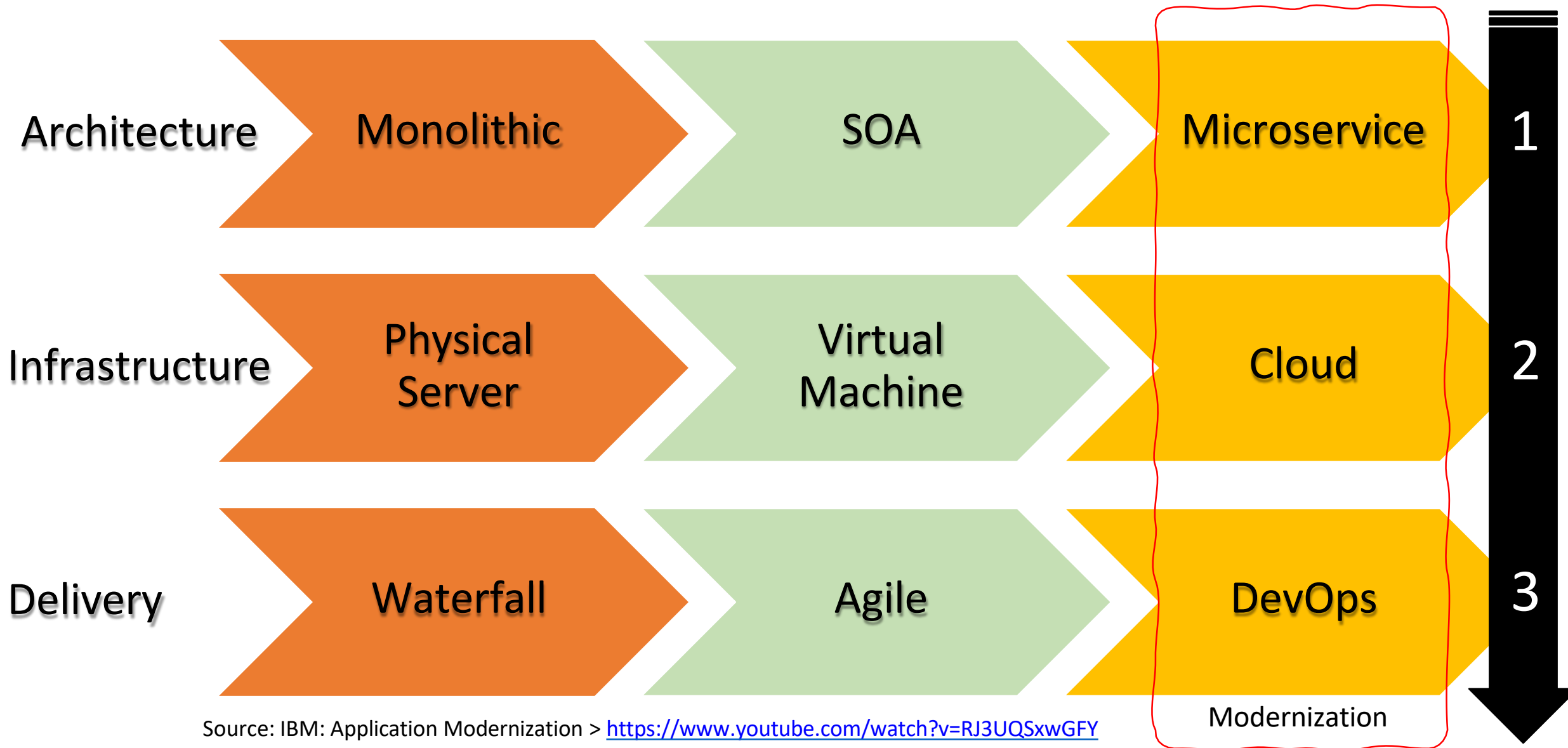
Kubernetes
Auditing

ML / AI

4

Slides are color coded based on the topic colors.

Application Modernization – 3 Transformations



Monolithic

Optional
Design
Patterns

Domain Driven Design

Event Sourcing and CQRS

Enterprise Service Bus

Relational Database [SQL] / NoSQL

Continuous Integration (CI)

Development

QA / QC

Ops

Waterfall

6/12 Months

Microservices

Domain Driven Design

Event Sourcing and CQRS

Event Streaming / Replicated Logs

Infrastructure Design Patterns

Container

Orchestrator

Service Mesh

SQL

NoSQL

CI

CD

DevOps

Agile

Scrum (4-6 Weeks)

Scrum / Kanban (1-5 Days)

Mandatory
Design
Patterns



Monitoring & Observability

- Challenges in Monitoring
- Monitoring Vs. Observability
- ML / AI – based Analytics

Challenges in Monitoring

Blind Spot

Container / Pod Disposability increases Portability and Scalability – However, this creates blind spots in Monitoring.

Need to Record

Portability of inter-dependent components creates an increased need to maintain and record telemetry data with traceability to ensure Observability.

Visualization

The scale and complexity introduced by the Containers and Container Orchestration good tools to Visualize and Analyze the data generated.

Don't Leave DevOps in Dark

Application performance is Critical for Ops Team as Containers can be scaled up and down in lightning speed.

Source: A Beginners guide to Kubernetes Monitoring by Splunk

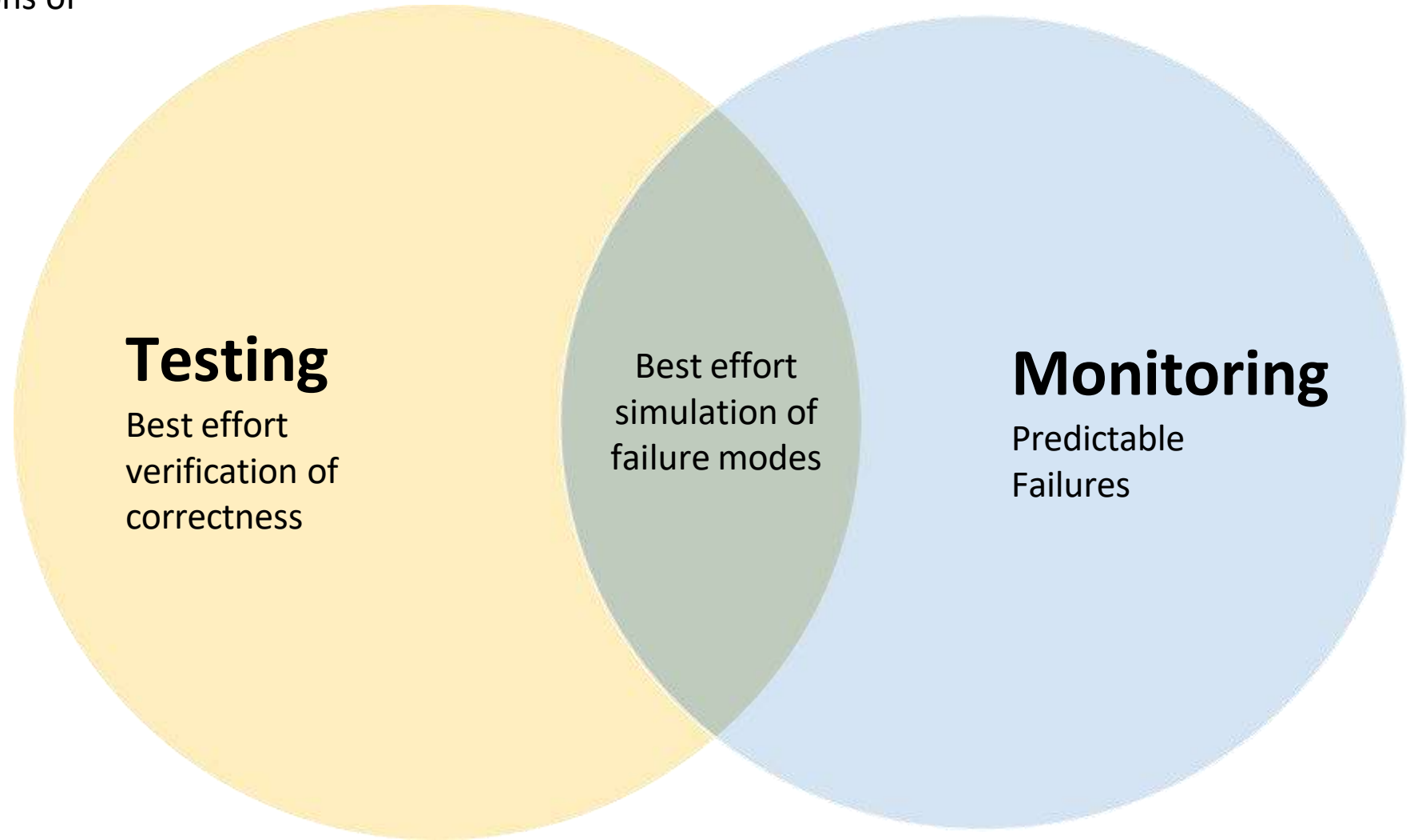
Monitoring Vs. Observability

	Monitoring	Observability
1	Says whether the System is Working or Not	Why its not working
2	Collects Metrics and Logs from a System	Actionable Insights gained from the Metrics
3	Failure Centric	Overall Behavior of the System
4	Is “the How” of something you do	Is “The Process” of something you have
5	I monitor you	You make yourself observable

Source: A Beginners guide to Observability by Splunk

Observability

All possible permutations of
full and partial failure



Source: A Beginners guide to Observability by Splunk

Benefits of Observability

1. Better understanding of complex microservices communication and end-user usage patterns
2. Helps in faster troubleshooting and shorter MTTR (Mean Time To Recovery)
3. Better understanding of incidents
4. Better uptime and performance
5. Happier customers and more revenue

Source: A Beginners guide to Observability by Splunk

Pillars of Observability

Logs/events



Immutable records of discrete events that happen over time

Metrics



Numbers describing a particular process or activity measured over intervals of time

Traces



Data that shows, for each invocation of each downstream service, which instance was called, which method within that instance was invoked, how the request performed, and what the results were

Source: A Beginners guide to Observability by Splunk

Events / Logs

Event Sources



- System and Server logs (syslog)
- Firewall and IDS/IPS logs
- Container / Pod Logs
- Application / Service / Database logs (log4j, log4net, Apache, MySQL, AWS)



- Infrastructure Metrics (Node, K8s)
- System Metrics (CPU, Memory, Disk)
- Service Metrics (Envoy Proxy)
- Network Metrics (Packets, Bytes)
- Business metrics (revenue, customer sign-ups, bounce rate, cart abandonment)
- UI Metrics (Google Analytics, Digital Experience Management)

Traces



- Specific parts of a user's journey are collected into traces, showing
- Which services were invoked,
- Which containers/hosts/instances they were running on, and
- what the results of each call were.



Kubernetes Auditing

- Auditing
- Audit Stages
- Audit Policy
- Audit Example

Kubernetes Auditing

Auditing Provides logs on what's happening within the cluster.
Scope and Levels of details are configurable

Forensics review of the Kubernetes logs shows the following

- **What** happened?
- **When** did it happen?
- **Who** initiated it?
- **On** what did it happen?
- **Where** was it observed?
- **From where** was it initiated?
- **To where** was it going?

Source: <https://kubernetes.io/docs/tasks/debug-application-cluster/audit/>

Kubernetes Audit Stages

Request Received

The stage for events generated as soon as the audit handler receives the request.

Response Started

Once the response headers are sent, but before the response body is sent.

Response Completed

The response body has been completed and no more bytes will be sent.

Panic

Events generated when a panic occurred.

Source: <https://kubernetes.io/docs/tasks/debug-application-cluster/audit/>

Lifecycle of an Audit Event



Kubernetes Audit Policy

None

Don't log events that match this rule.

MetaData

Log request metadata (requesting user, timestamp, resource, verb, etc.) but not request or response body.

Request

Log event metadata and request body but not response body. This does not apply for non-resource requests.

Request Response

Log event metadata, request and response bodies. This does not apply for non-resource requests.

Source: <https://kubernetes.io/docs/tasks/debug-application-cluster/audit/>

Kubernetes Audit Policy Example

```
! Audit-Policy-Example.yaml X
! Audit-Policy-Example.yaml > apiVersion
1  apiVersion: audit.k8s.io/v1 # This is required.
2  kind: Policy
3  # Don't generate audit events for all requests in RequestReceived stage.
4  omitStages:
5  | - "RequestReceived"
6  rules:
7  | # Log pod changes at RequestResponse level
8  | - level: RequestResponse
9  |   resources:
10 |     - group: ""
11 |       # Resource "pods" doesn't match requests to any subresource of pods,
12 |       # which is consistent with the RBAC policy.
13 |       resources: ["pods"]
14 |   # Log "pods/log", "pods/status" at Metadata level
15 |   - level: Metadata
16 |     resources:
17 |       - group: ""
18 |         resources: ["pods/log", "pods/status"]
19
20 |   # Don't log requests to a configmap called "controller-leader"
21 |   - level: None
22 |     resources:
23 |       - group: ""
24 |         resources: ["configmaps"]
25 |         resourceNames: ["controller-leader"]
26
27 |   # Don't log watch requests by the "system:kube-proxy" on endpoints or services.
28 |   - level: None
29 |     users: ["system:kube-proxy"]
30 |     verbs: ["watch"]
31 |     resources:
32 |       - group: "" # core API group
33 |         resources: ["endpoints", "services"]
34
35 |   # Don't log authenticated requests to certain non-resource URL paths.
36 |   - level: None
37 |     userGroups: ["system:authenticated"]
38 |     nonResourceURLs:
39 |       - "/api*" # Wildcard matching.
40 |       - "/version"
```

Kubernetes Native Monitoring

Application Logs (L7 Logs)

Container / Pod Logs

- Process
- System Calls
- Network Logs
- File System Logs

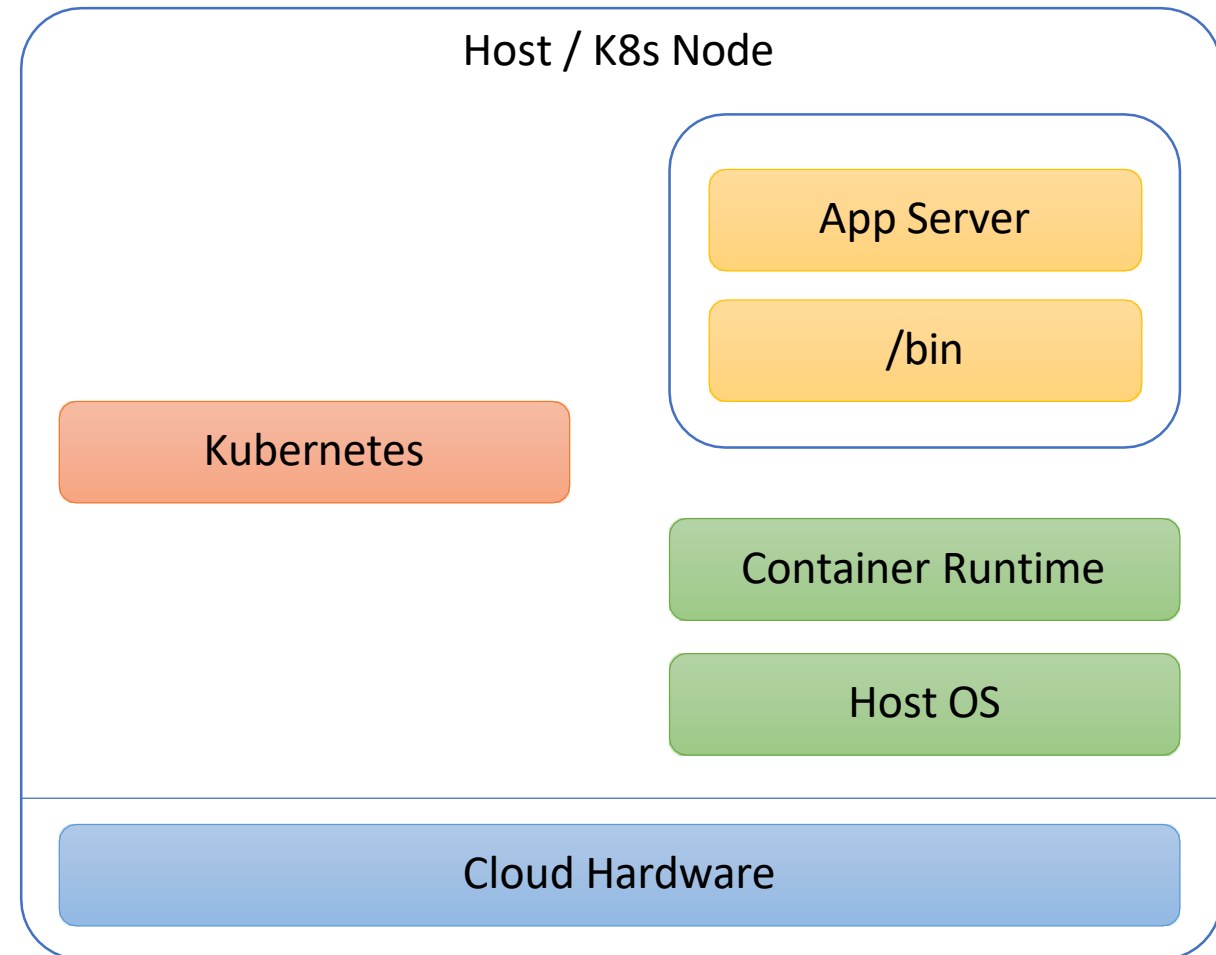
Kubernetes Logs

- Network Flow Logs
- Audit Logs
- DNS Logs

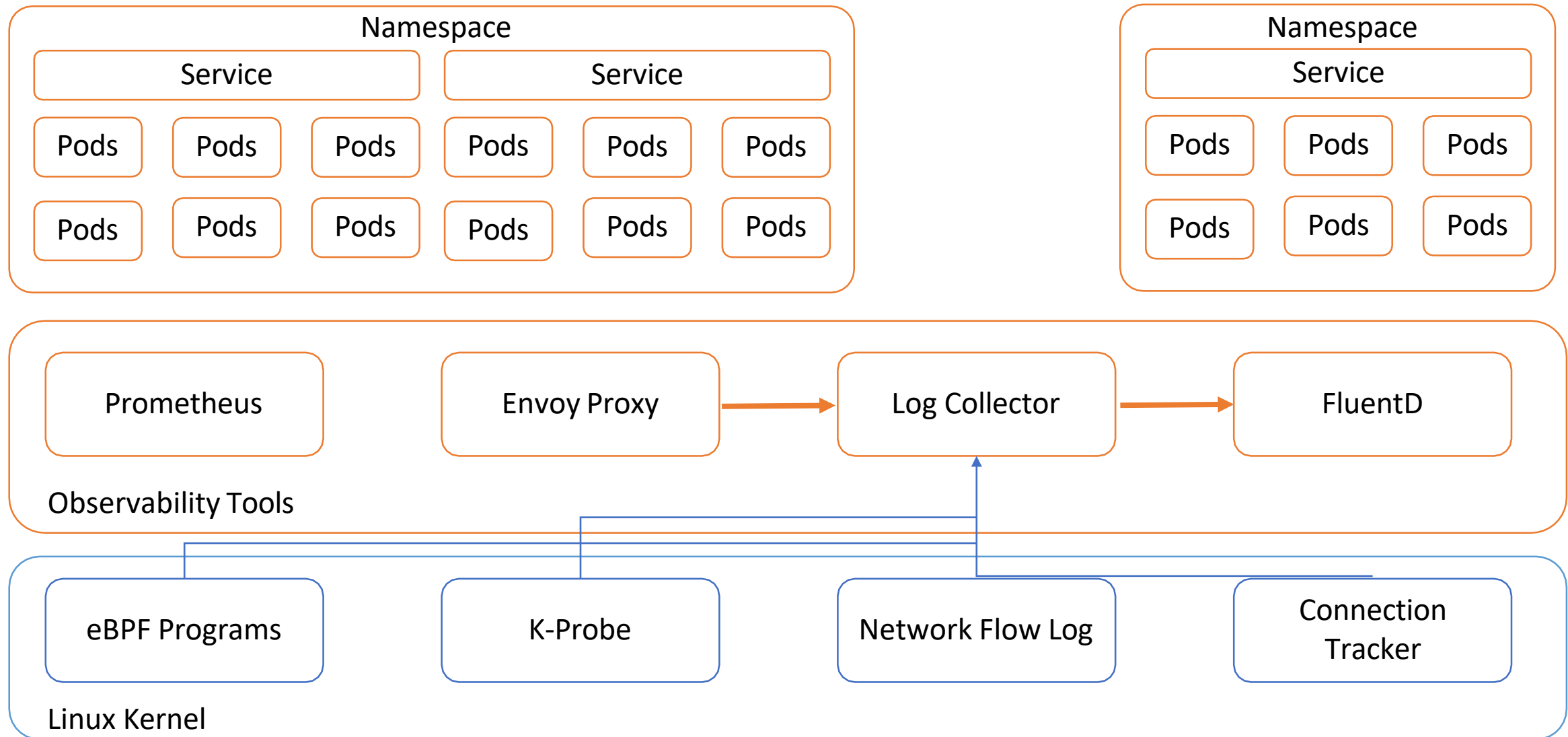
Host OS Logs

- SSH Logs
- OS Audit Logs

Cloud Infra Logs



Kubernetes Node



Data Collection

K-Probe

Source IP Address, Source Port, Destination IP Address, Destination Port, Protocol

NF Log

Adds Bytes and Packets count for the above five attributes for a connection

Log Collector

Adds Kubernetes Meta Data to the above data like Namespace, Service, Pod etc..

Prometheus

Collects metrics, System, Service metrics

Kubernetes Metrics Server

- **Metrics Server** is a cluster-wide aggregator of resource usage data.
- CPU is reported as the average usage, in CPU cores, over a period of time.
- Memory is reported as the working set, in bytes, at the instant the metric was collected.

Source: <https://kubernetes.io/docs/tasks/debug-application-cluster/resource-metrics-pipeline/>



Zipkin / Prometheus / Grafana / Kiali



General / MetaArivu Microservice Template ★ 🔗



🕒 Last 15 minutes ▾



Basic Statistics

Uptime

41.5 min

Start time

an hour ago

Heap Used

4.0%

Non-Heap Used

4.9%

Process Open Files



CPU Usage



Load Average



JVM Statistics - Memory

PS Eden Space (heap)



PS Old Gen (heap)



PS Survivor Space (heap)



@arafkarsh



arafkarsh

Shopping Portal App

Sign In
To Shopping Portal App

Login Id *

Please enter valid login id

Password *


LOG IN

Create Account

Shopping Portal Ketan Gote

Catalog My Cart My Orders


iPhone 12 Pro \$ 79000



iPhone 12 Pro

Reviews ADD TO CART

iPhone 13 Pro \$ 79000









iPhone 13 Pro

Reviews ADD TO CART

Shopping Portal Ketan Gote

Catalog My Cart My Orders

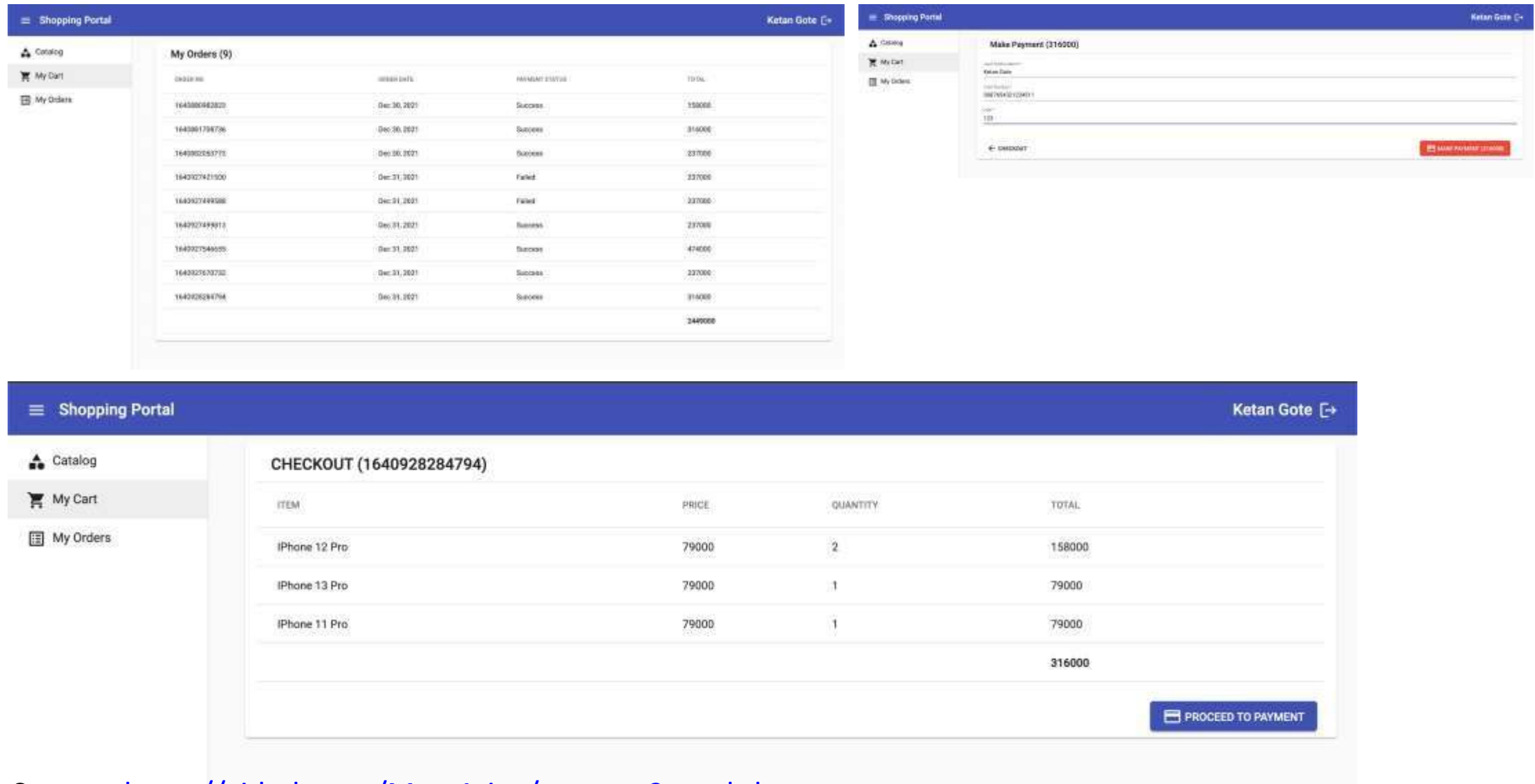
My Cart (3)

ITEM	PRICE	QUANTITY	TOTAL	Action
iPhone 12 Pro	79000	2	158000	 
iPhone 13 Pro	79000	1	79000	 
iPhone 11 Pro	79000	1	79000	 
			316000	

CHECKOUT CART

Source: <https://github.com/MetaArivu/ecommerce-workshop>

Shopping Portal App



Source: <https://github.com/MetaArivu/ecommerce-workshop>

Zipkin

The screenshot shows the Zipkin web interface. At the top, there's a navigation bar with the Zipkin logo, a search bar labeled "Find a trace", a "Dependencies" link, a language selector set to "ENGLISH", and a "Search by trace ID" input field. Below the navigation bar, there's a search bar with a red "+" button, a "RUN QUERY" button, and a settings gear icon. The main content area displays "10 Results" and buttons for "EXPAND ALL", "COLLAPSE ALL", and "Service filters". A table lists the traces with columns for "Root", "Start Time", "Spans", and "Duration". Each row includes a dropdown arrow, the service name and operation, the start time, the number of spans, a duration bar, the duration value, and a "SHOW" button.

Root	Start Time	Spans	Duration
shopping-cart-enhancer-svc: poll	a few seconds ago (12/31 10:54:44:464)	4	333.691ms
payment-service: send	a few seconds ago (12/31 10:55:16:248)	8	268.475ms
order-service: get /api/v1/checkout	a few seconds ago (12/31 10:54:45:749)	1	21.169ms
payment-service: post /api/v1/	a few seconds ago (12/31 10:55:16:234)	1	18.388ms
user-svc: get /user-svc/live	a few seconds ago (12/31 10:55:01:887)	1	7.354ms
user-svc: get /user-svc/ready	a few seconds ago (12/31 10:54:46:888)	1	2.610ms
user-svc: get /user-svc/live	a few seconds ago (12/31 10:55:16:888)	1	2.201ms
user-svc: get /user-svc/ready	a few seconds ago (12/31 10:55:16:888)	1	1.856ms

Source: <https://github.com/MetaArivu/ecommerce-3-workshop>

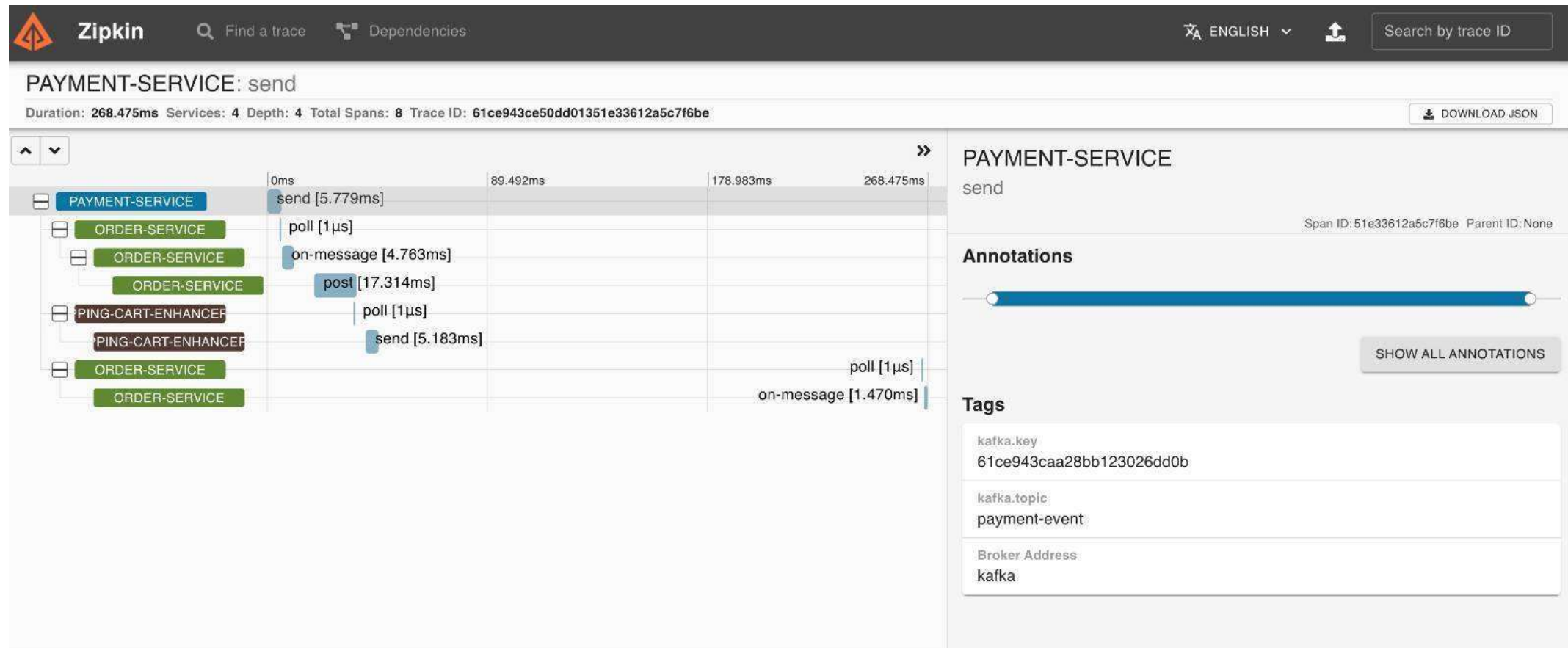
Zipkin Traces

The screenshot shows the Zipkin web interface. At the top, there's a navigation bar with the Zipkin logo, a search bar labeled 'Find a trace', a 'Dependencies' link, a language dropdown set to 'ENGLISH', and a 'Search by trace ID' button. Below the navigation bar is a search input field with a red '+' icon, a 'RUN QUERY' button, and a settings gear icon. The main content area displays '10 Results' and buttons for 'EXPAND ALL' and 'COLLAPSE ALL'. A 'Service filters' dropdown is also present. The table of results has columns for 'Root', 'Start Time', 'Spans', and 'Duration'. The first three rows are expanded, showing details for 'shopping-cart-enhancer-svc: poll', 'payment-service: send', and 'order-service: get /api/v1/checkout'. The 'payment-service: send' row includes a 'Trace ID' and a horizontal bar chart showing the sequence of services in the trace: order-service (5), kafka (5), shopping-cart-enhancer-svc (2), and payment-service (1).

Root	Start Time	Spans	Duration
shopping-cart-enhancer-svc: poll	a minute ago (12/31 10:54:44:464)	4	333.691ms
payment-service: send	a few seconds ago (12/31 10:55:16:248)	8	268.475ms
order-service: get /api/v1/checkout	a minute ago (12/31 10:54:45:749)	1	21.169ms

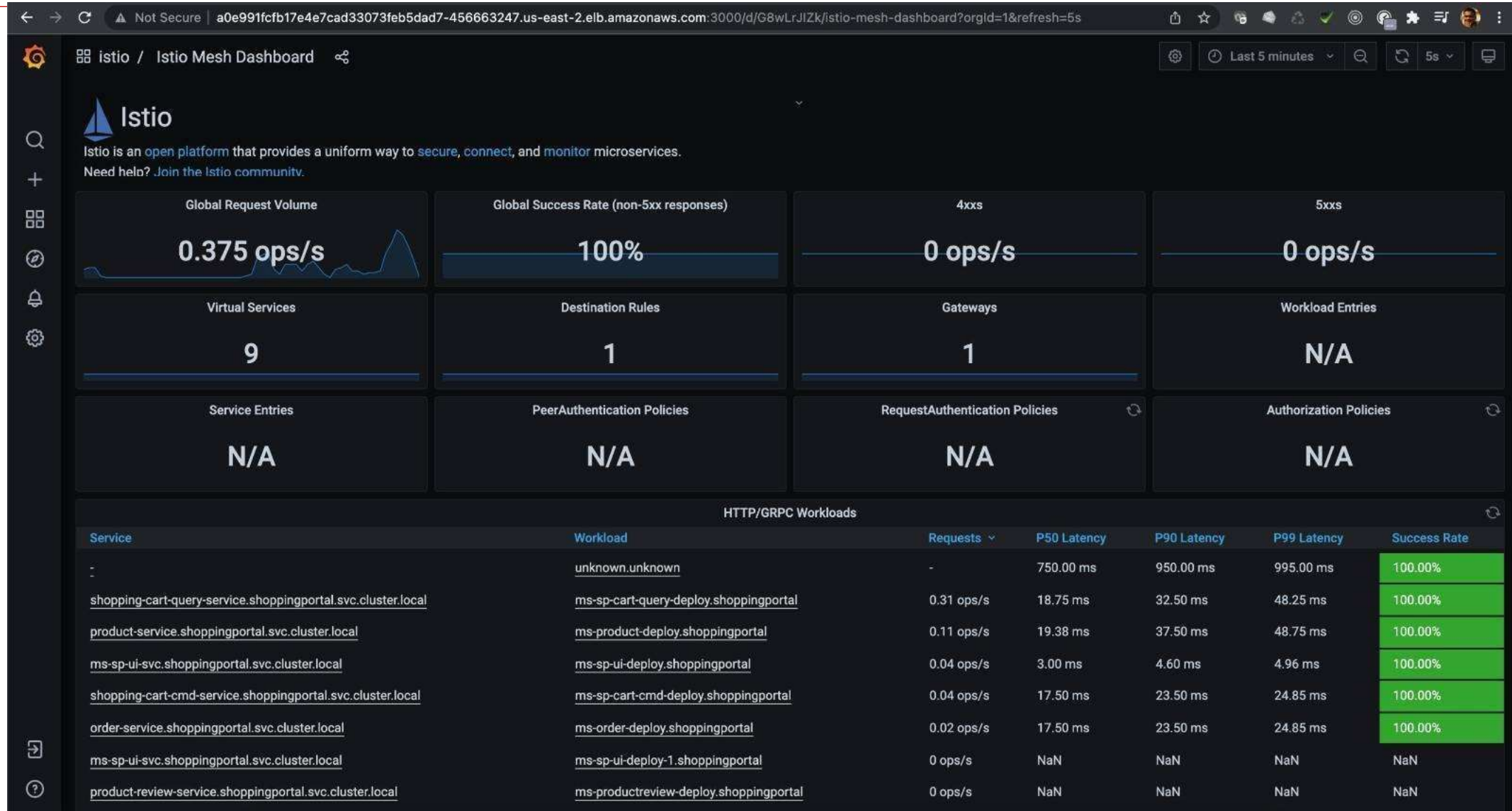
Source: <https://github.com/MetaArivu/ecommerce-3-workshop>

Zipkin Traces



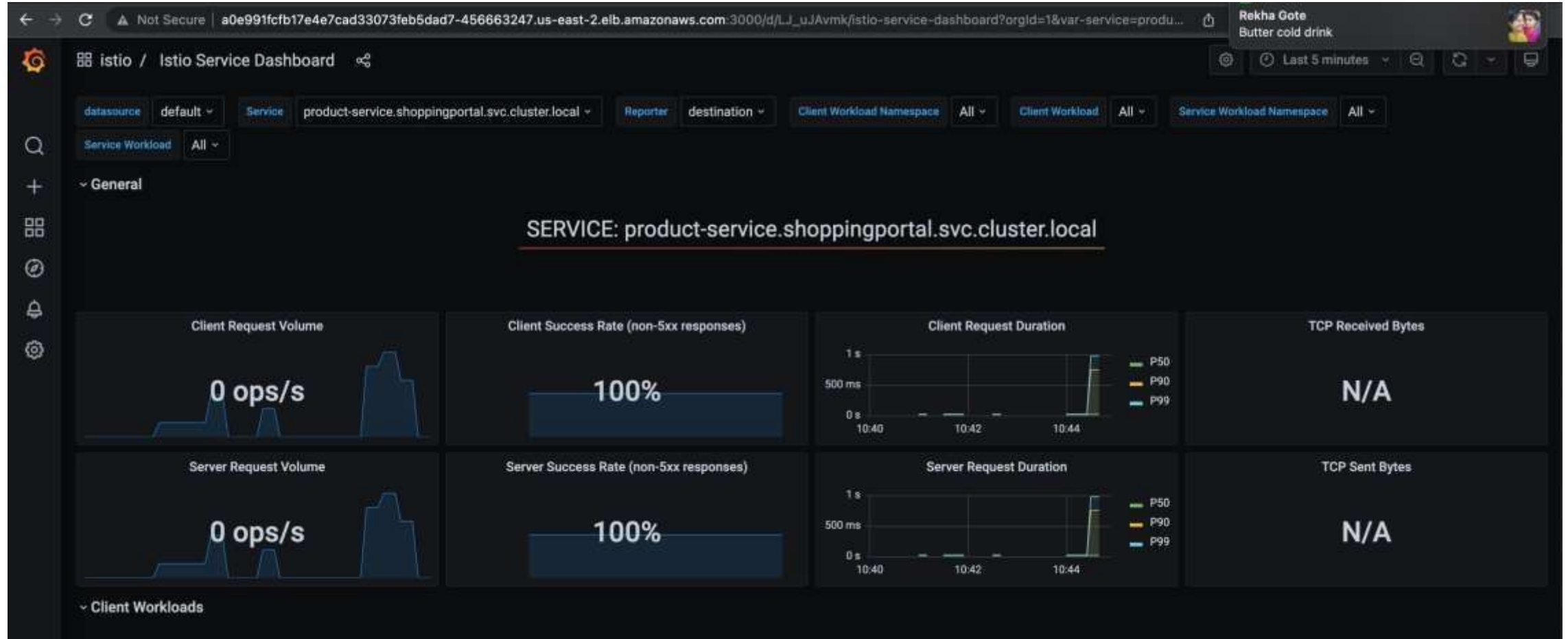
Source: <https://github.com/MetaArivu/ecommm-3-workshop>

Prometheus / Grafana



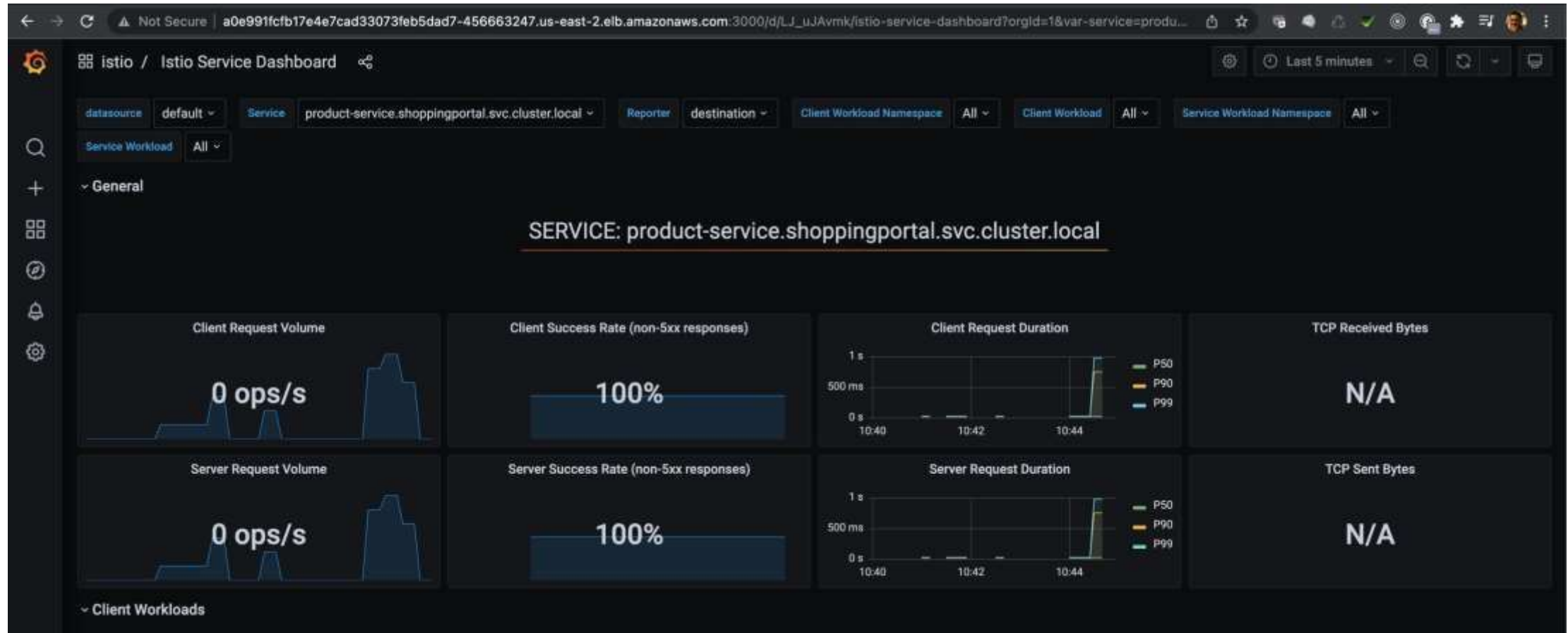
Source: <https://github.com/MetaArivu/ecommerce-3-workshop>

Prometheus / Grafana



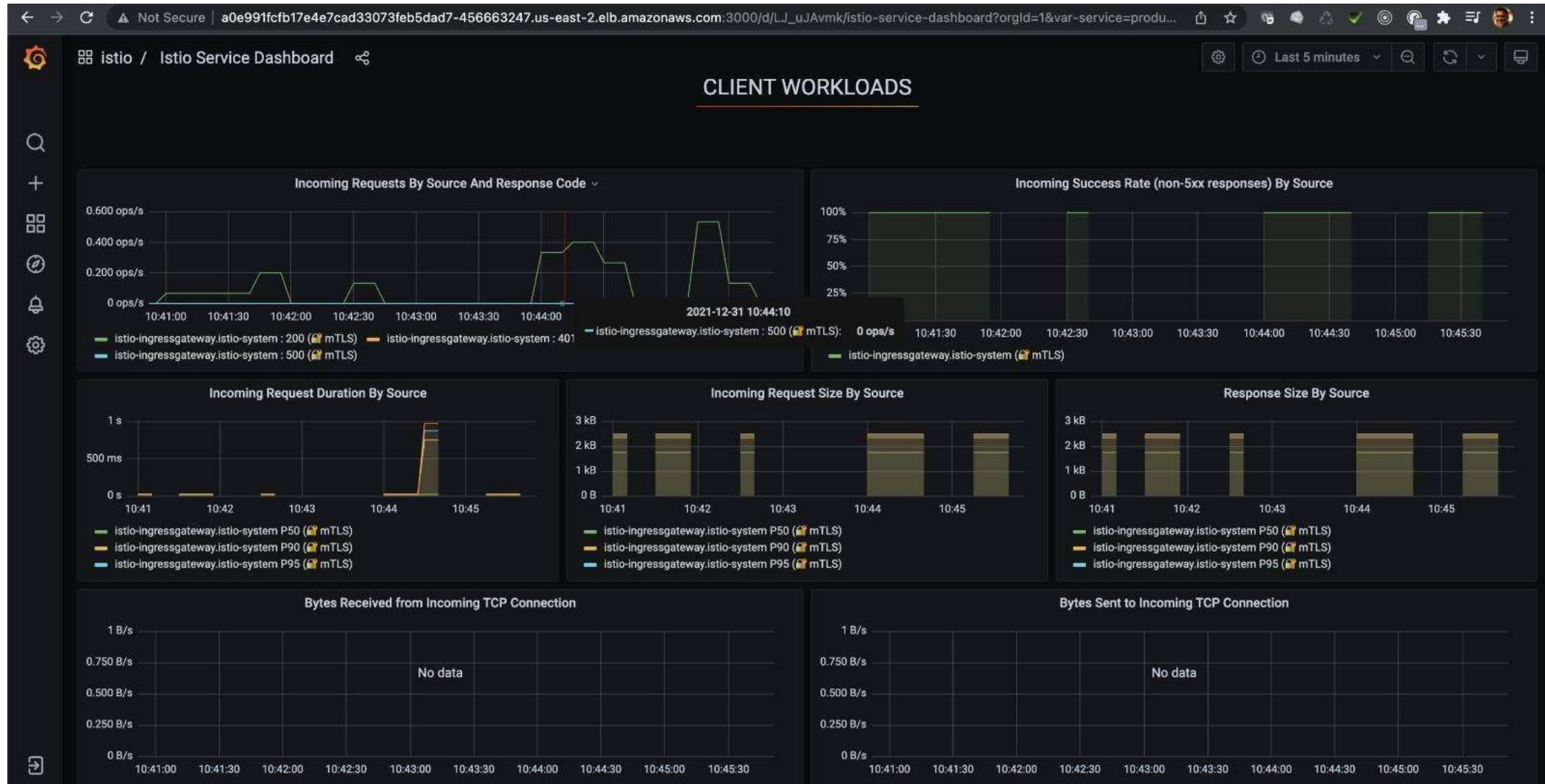
Source: <https://github.com/MetaArivu/ecommerce-3-workshop>

Prometheus / Grafana



Source: <https://github.com/MetaArivu/ecommerce-3-workshop>

Prometheus / Grafana



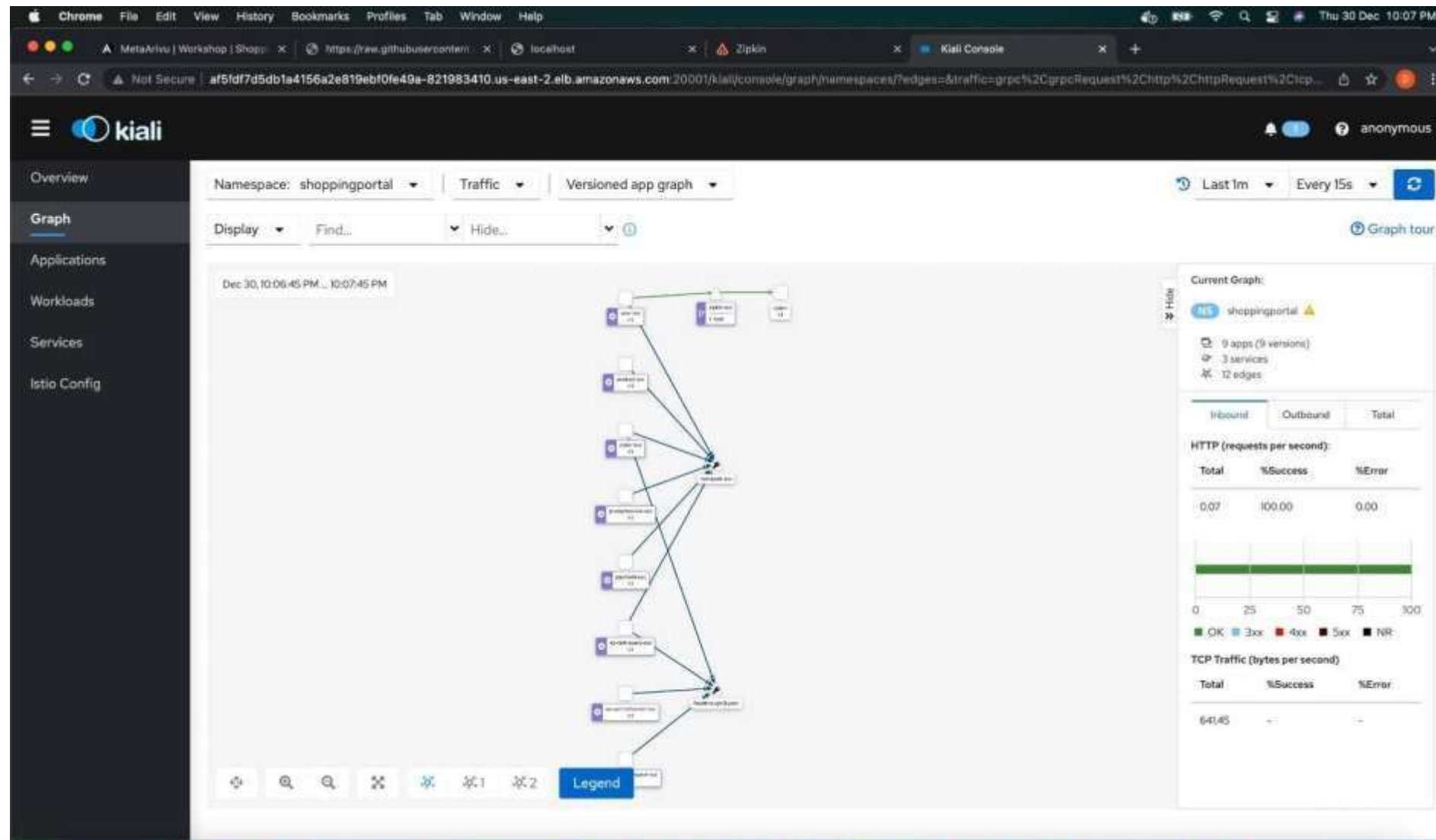
Source: <https://github.com/MetaArivu/ecommerce-3-workshop>

Prometheus / Grafana



Source: <https://github.com/MetaArivu/ecommerce-3-workshop>

Kiali



Source: <https://github.com/MetaArivu/ecommerce-3-workshop>

Kiali

The screenshot shows the Kiali console interface in a web browser. The left sidebar contains navigation links: Overview, Graph, Applications (selected), Workloads, Services, and Istio Config. The main panel displays the 'Applications' view for the 'shoppingportal' namespace. At the top, there's a dropdown for 'Namespace: shoppingportal' and buttons for 'Last 1m', 'Every 15s', and a refresh icon. Below this is a search bar 'App Name' and a filter 'Filter by App Name'. The applications are listed in a table with columns: Health, Name, Namespace, Labels, and Details. Four applications are shown: order-svc, payment-svc, product-svc, and productreview-svc. Each application has a green health status icon, a blue 'A' icon, and a 'VS' icon in the details column. The labels for each application are: app: order-svc, managed-by: m2, name: order-svc, release: stable, tier: fe, version: v1, zone: prod.

Health	Name	Namespace	Labels	Details
✓	order-svc	shoppingportal	app: order-svc, managed-by: m2, name: order-svc, release: stable, tier: fe, version: v1, zone: prod	shoppingportal
✓	payment-svc	shoppingportal	app: payment-svc, managed-by: m2, name: payment-svc, release: stable, tier: fe, version: v1, zone: prod	shoppingportal
✓	product-svc	shoppingportal	app: product-svc, managed-by: m2, name: product-svc, release: stable, tier: fe, version: v1, zone: prod	shoppingportal
✓	productreview-svc	shoppingportal	app: productreview-svc, managed-by: m2, name: productreview-svc, release: stable, tier: fe, version: v1, zone: prod	shoppingportal

Source: <https://github.com/MetaArivu/ecommerce-3-workshop>



ML / AI

- Analytics
- Anomalous Events Example

ML/AI Driven Analytics

- **Enrich:** Adding context to events to make them informative and actionable
- **Reduce Duplicate:** Automatically concealing duplicate events to focus on relevant ones and reducing alert storms
- **Reduce False +ve:** Reducing event clutter and false positives with multivariate anomaly detection
- **Filter/Tag/Sort:** Easily sifting through vast amounts of events by filtering, tagging and sorting

Source: A Beginners guide to Observability by Splunk

Anomalous Events

IP Sweep Detection

Pods sending many packets to many destinations

Port Scan Detection

Pods sending packets to One Destination on multiple ports.

HTTP Spike

Service that get too many HTTP inbound Connections


DNS Latency

Too High Latency for DNS Requests

L7 Latency


Pods with Too High Latency for L7 Requests

Source: Kubernetes Security and Observability: Brendan Creane & Amit Gupta

 **cicd-github-action** lines 816

Public

CI / CD Pipeline using Git Actions. Fully automated pipeline with continuous deployment up to the Production Environment. CI / CD Pipeline using Jenkins and Tekton available in other repositories.

 Java  2

 **terraform-quickstart**

lines 11.8k Public

Terraform Examples. How to create your cloud infrastructures, examples include AWS VPC, EC2, S3, Route53, ELB, EKS, Apache Web Server, RDS (MySQL), MongoDB, Redis, Apache Solr, Kafka, etc.

 3

 **k8s-quickstart** lines 373.5k

Public

Kubernetes in Docker (KinD) Examples with 3 Node clusters and 5 Node Cluster and deploying a Web Server and 2 Tomcats as App Servers and an E-Commerce App with Order, Cart, and Payment Services.

 HTML  3

 **Kafka-quickstart** lines 19.8k

Public

Kafka Examples focusing on Producer, Consumer, KStreams, KTable, Global KTable using Spring, Kafka Cluster Setup & Monitoring. Implementing Event Sourcing and CQRS Design Pattern using Kafka

 Java  29  12

 **ms-quickstart** lines 5.6k

Public template



Microservice Template gives you a SpringBoot App template with Open API 3 Ex, Actuator, Sleuth, Prometheus, and POM file with Fat and Thin jar file creation and Dockerfile for containerization.

 Java  1  2

 **ms-test-quickstart**

lines 63.5k Public

Microservice Test Automation Example for SpringBootTest 2, JUnit 5, Cucumber 6, Selenium 4, Mockito 3, WireMock 2, Pact 4. All the testing frameworks are required to fully automate the testing for ...

 Java  1

Source Code: <https://github.com/MetaArivu>

Web Site: <https://metarivu.com/>

<https://pyxida.cloud/>

Thank you

DREAM | AUTOMATE | EMPOWER

Araf Karsh Hamid :

India: +91.999.545.8627

<http://www.slideshare.net/arafkarsh>

<https://www.linkedin.com/in/arafkarsh/>

<https://www.youtube.com/user/arafkarsh/playlists>

<http://www.arafkarsh.com/>

@arafkarsh



arafkarsh



@arafkarsh



arafkarsh