



# Application Performance Monitoring

Build Better Customer Experiences with AWS Observability

Surbhi Dangi

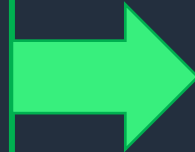
Product and Design Lead, [surbd@amazon.com](mailto:surbd@amazon.com)

# Agenda

1. Monitoring more than failures
2. Real user monitoring
3. Synthetic monitoring
4. Internet monitoring
5. Tracing
6. Helpful resources

# How do you measure what good looks like to your **end users**

- .
- .
- Failed payments*
- JavaScript errors*
- Missing UX*
- Slow loading web pages*
- Pop-overs covering your CTA*
- .
- .



**Performance  
&  
Availability**

# Monitoring more than failures



---

Is it behaving  
as expected?



---

What is the usage?



---

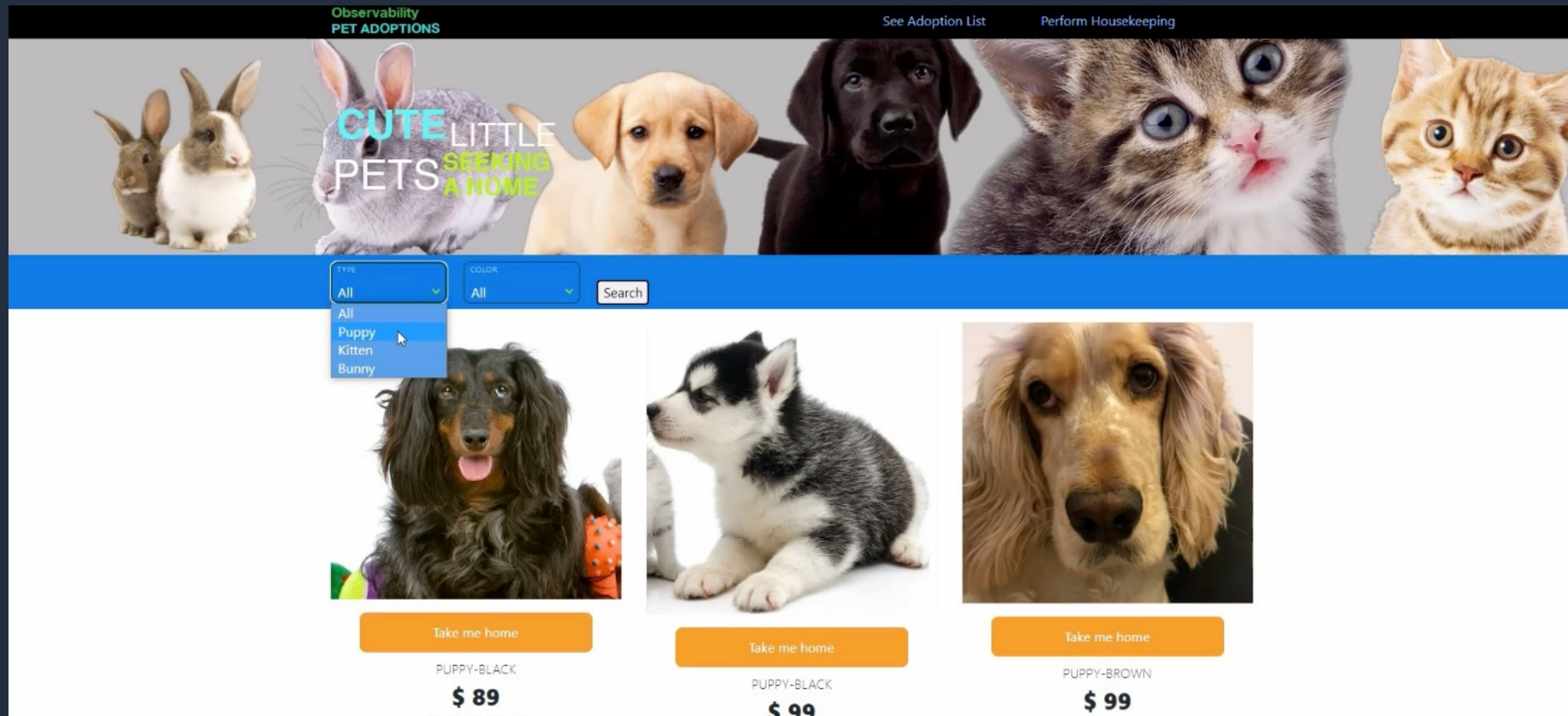
What is the  
business impact?

## REAL USER MONITORING

# Measure the performance of real users interacting with your application

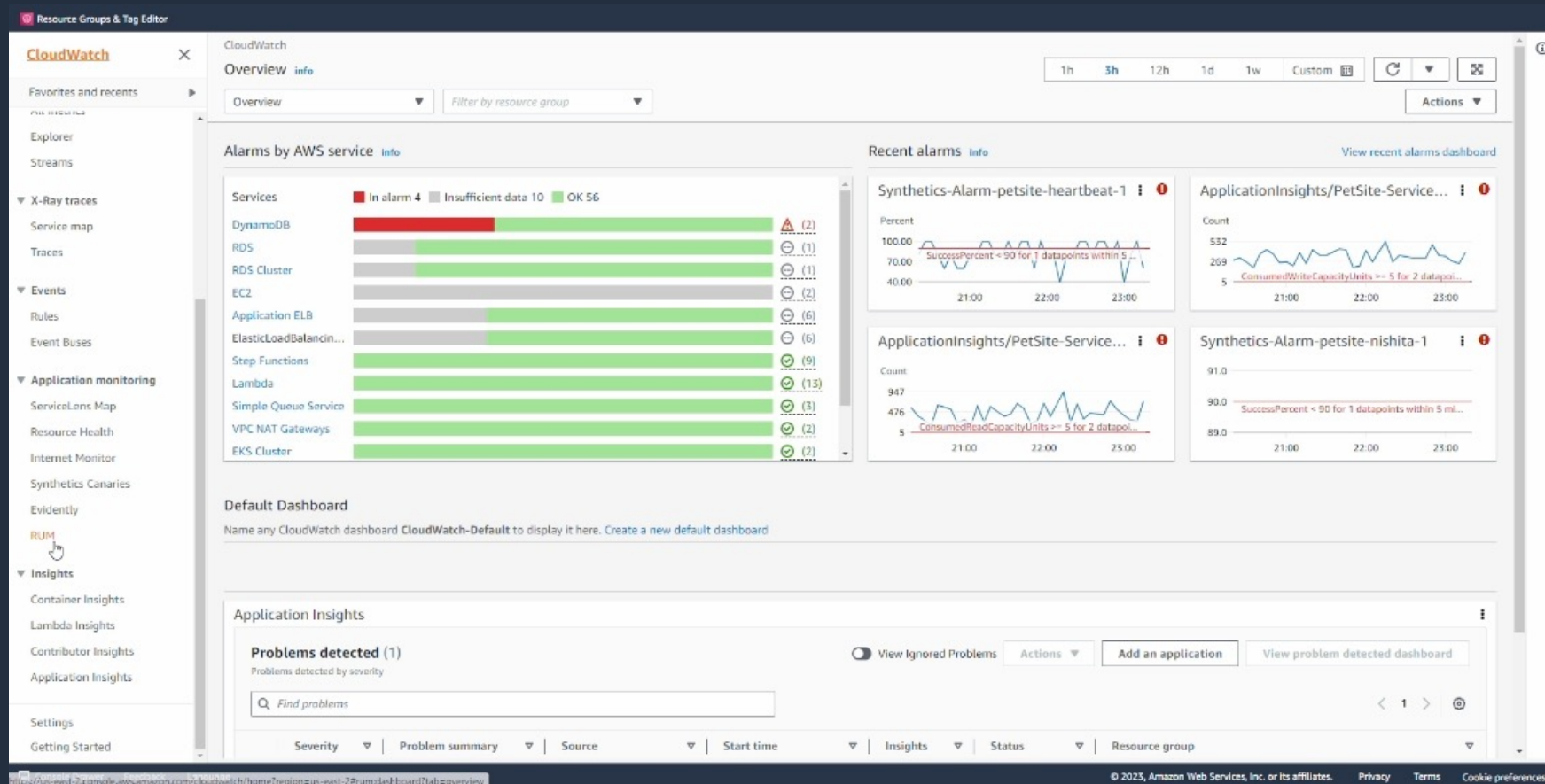
- Identify performance issues before too many users experience them
- Understand what is causing “slowness” and errors; fixing them proactively
- Analyze customer journeys and understand where customers spend most time

# Pet adoption site!



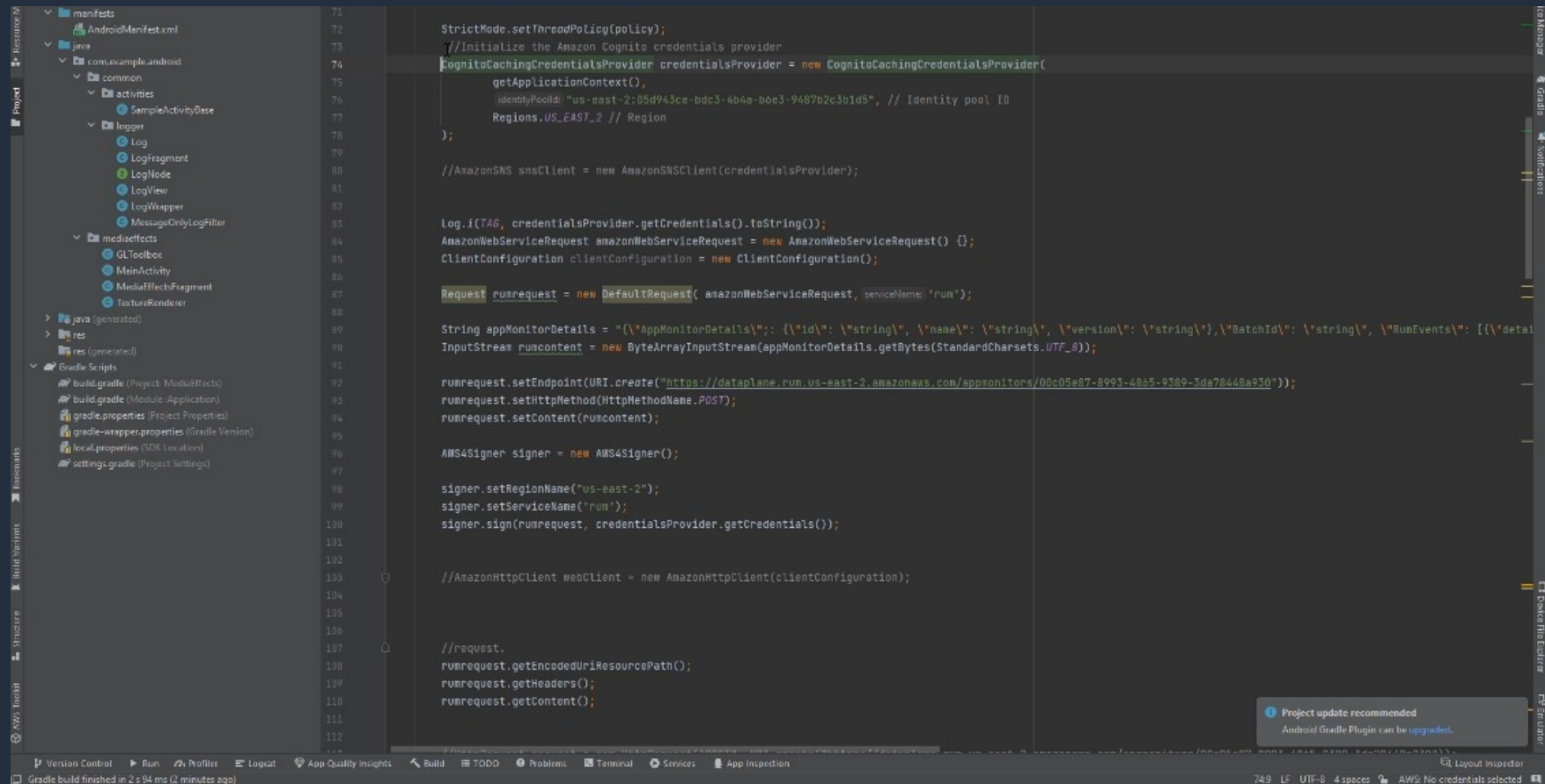
## AMAZON CLOUDWATCH RUM

# Understand what real users experience in the application



## AMAZON CLOUDWATCH RUM

# Monitor your mobile experiences



```
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112
```

```
StrictMode.setThreadPolicy(policy);  
//Initialize the Amazon Cognito credentials provider  
CognitoCachingCredentialsProvider credentialsProvider = new CognitoCachingCredentialsProvider(  
    getApplicationContext(),  
    IdentityPoolId: "us-east-2:05d943ce-bdc3-4b4a-b6e3-9487b2c3b1d5", // Identity pool ID  
    Regions.US_EAST_2 // Region  
);  
  
//AmazonSNS snsClient = new AmazonSNSClient(credentialsProvider);  
  
Log.i(TAG, credentialsProvider.getCredentials().toString());  
AmazonWebServiceRequest amazonWebServiceRequest = new AmazonWebServiceRequest() {};  
ClientConfiguration clientConfiguration = new ClientConfiguration();  
  
Request rumrequest = new DefaultRequest(amazonWebServiceRequest, "service-name: rum");  
  
String appMonitorDetails = "{ \"id\": \"\", \"name\": \"\", \"version\": \"\", \"BatchId\": \"\", \"RunEvents\": [ { \"detail\"  
InputStream rumcontent = new ByteArrayInputStream(appMonitorDetails.getBytes(StandardCharsets.UTF_8));  
  
rumrequest.setEndpoint(URI.create(\"https://dataplane.rum.us-east-2.amazonaws.com/appmonitors/00c05e87-8993-4845-9389-3da78448a930\"));  
rumrequest.setHttpMethod(HttpMethodName.POST);  
rumrequest.setContent(rumcontent);  
  
AWS4Signer signer = new AWS4Signer();  
  
signer.setRegionName(\"us-east-2\");  
signer.setServiceName(\"rum\");  
signer.sign(rumrequest, credentialsProvider.getCredentials());  
  
//AmazonHttpClient webClient = new AmazonHttpClient(clientConfiguration);  
  
//request.  
rumrequest.getEncodedUriResourcePath();  
rumrequest.getHeaders();  
rumrequest.getContent();
```

Project update recommended  
Android Gradle Plugin can be upgraded.

749 LF UTF-8 4 spaces AWS No credentials selected





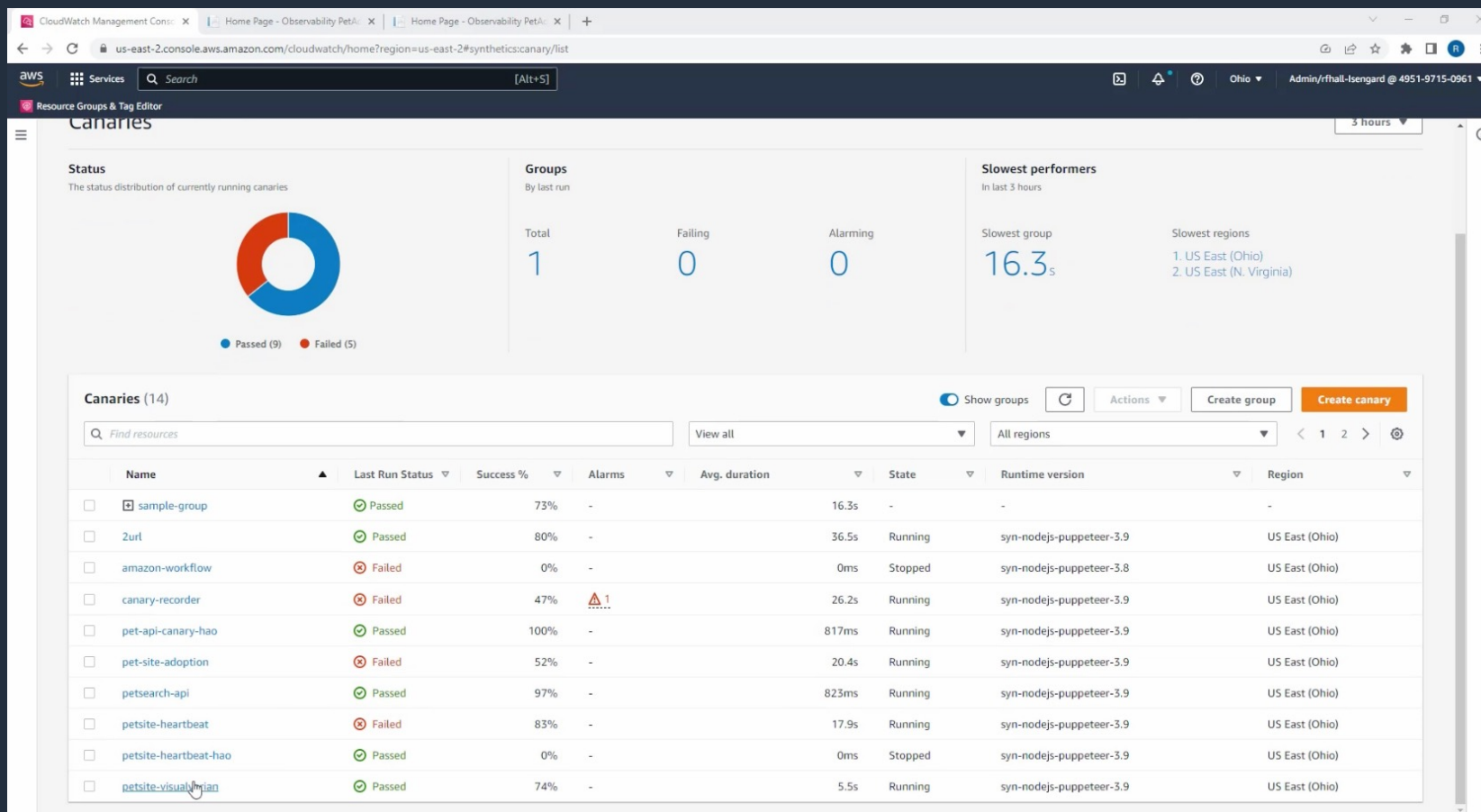
## SYNTHETIC MONITORING

# Identify performance problems in your website or application by simulating user interactions

- Simulate user experiences to identify problems before they affect users
- Powerful tool for Support teams to get an idea of what end users might be experiencing
- Monitor workflows that are not heavily trafficked yet critical

# AMAZON CLOUDWATCH SYNTHETICS

## Identify unexpected behavior before your users do



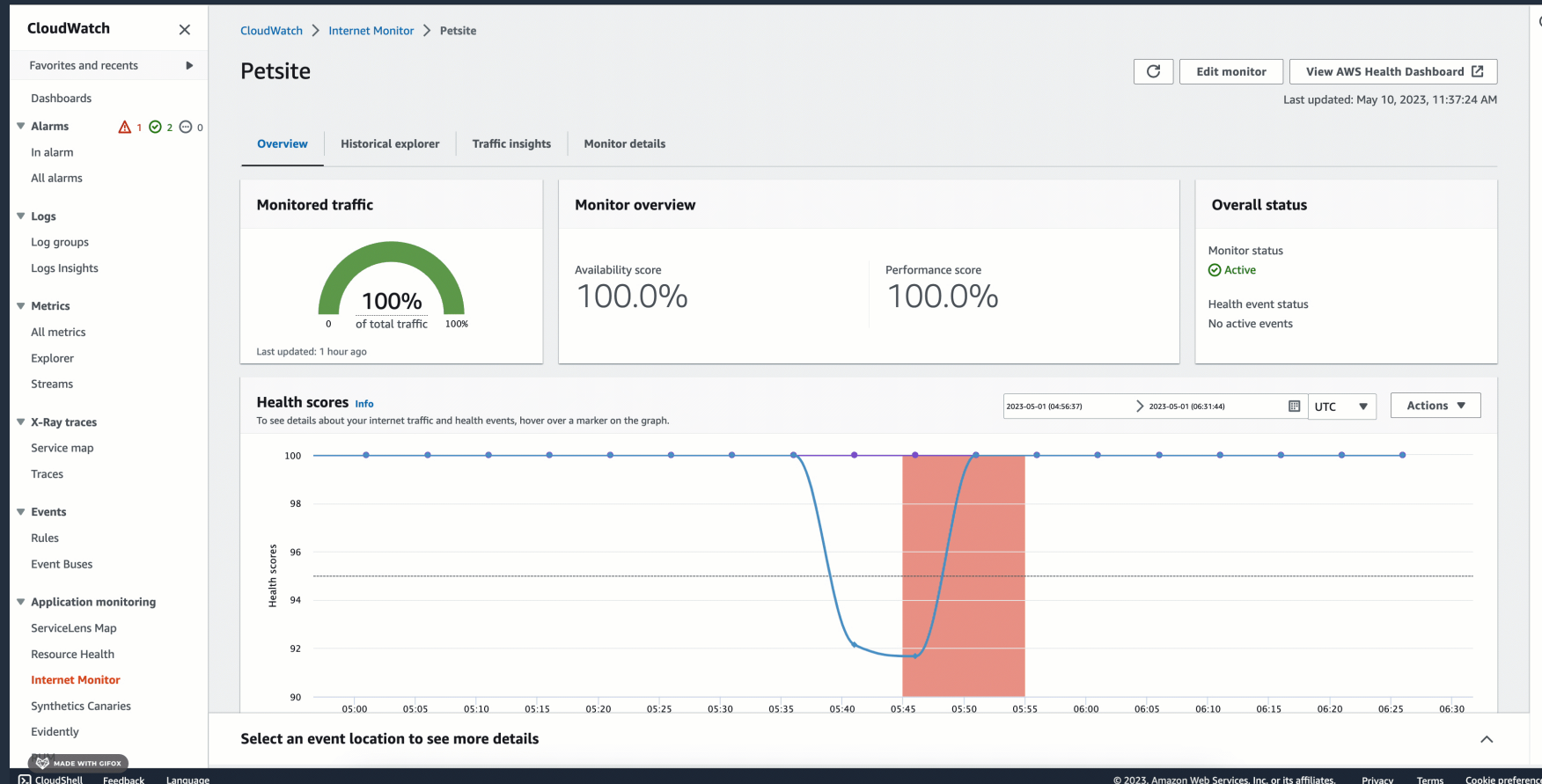
## INTERNET MONITORING

# Identify performance problems in your website or application by simulating user interactions

- Identify which customer location(s) or ISP(s) are impacting your customer experience
- Understand the source of latencies or performance degradation quickly
- Powerful tool for Support teams for quicker root cause

# CLOUDWATCH INTERNET MONITOR

## Path of the internet between your users and your application



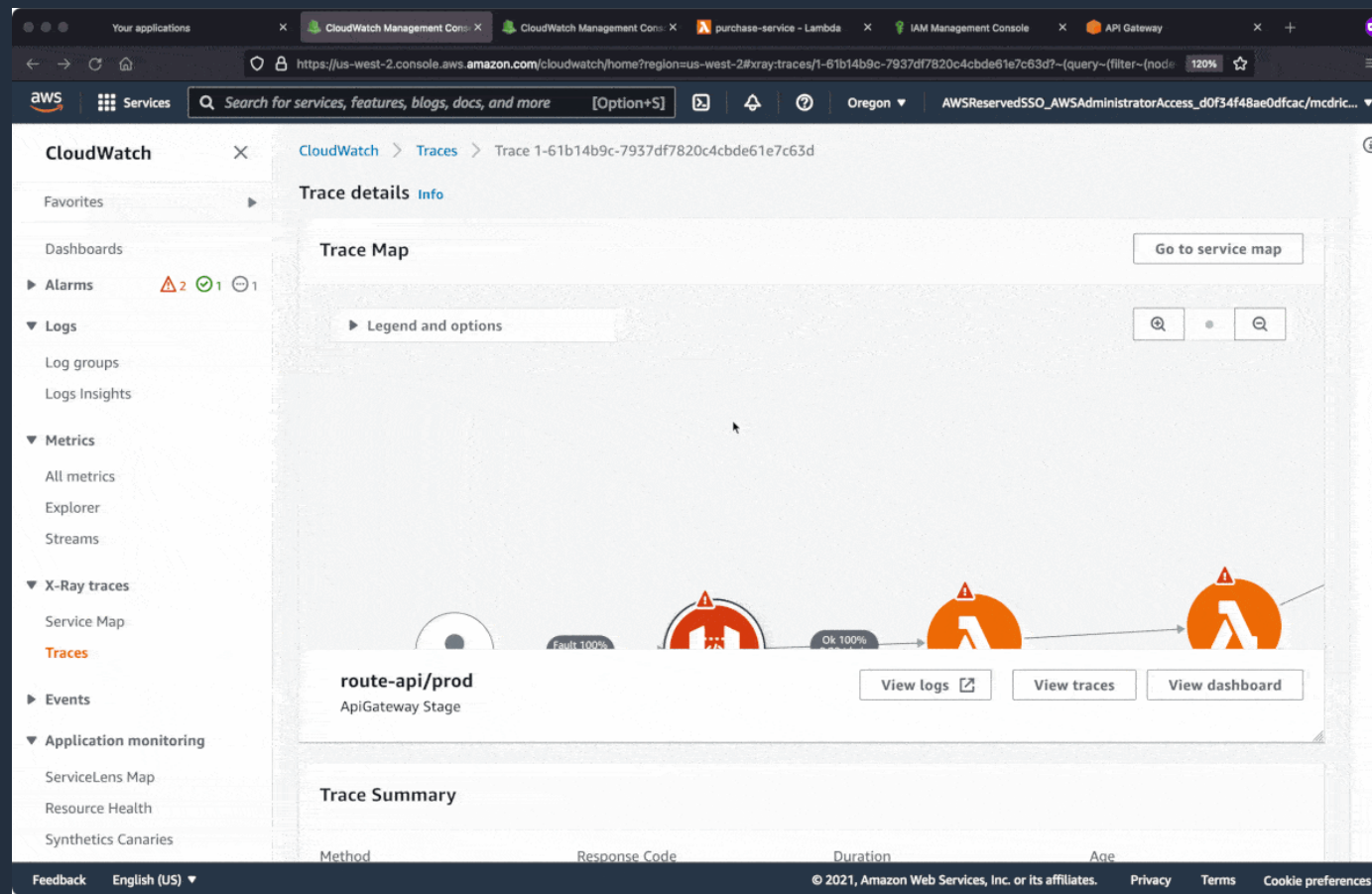
## TRACING REQUESTS

# Connect the dots from application to infrastructure metrics, network calls, and live processes

- Quickly find individual requests or traces of interest for root cause analysis
- Understand the source of latencies or performance degradation
- Determine whether the issue is an anomaly, a transient spike, or persistent

## TRACING USING AWS X-RAY

# Trace request from the customer to the infrastructure



# Get started with these self-guided workshops

One Observability  
Workshop



AWS Observability  
Best Practices



Skill Builder – AWS  
Observability



## Coming up next...

- Build better customer experiences with observability
- Get actionable insights using Log Analytics
- Curated container observability experiences
- Monitor modern applications - The cloud-native way
- Monitor modern applications - The managed open-source way





# Thank you!

Surbhi Dangi

surbd@amazon.com