

BEAM  
SUMMIT

# Scaling public internet data collection with Apache Beam



# Agenda



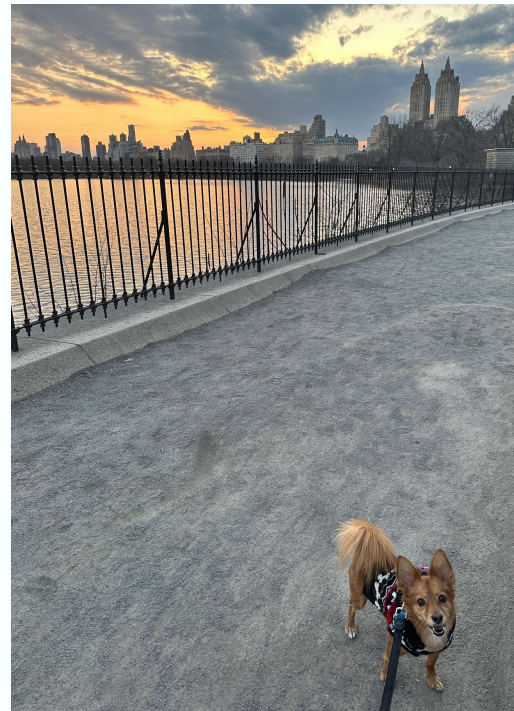
- Cortex Xpanse, Palo Alto Networks - Overview
- Beam @ Xpanse
- Our Beam Guidelines
- A Performance Tuning case



# Hi there



- Lior Dadosh
- Sr. Software Engineer @ Palo Alto Networks
- Based in New York



Cortex Xpanse

# Attack Surface Management:

"The process of continuously discovering, identifying, inventorying, and assessing the exposures of an entity's IT asset estate."



# Attack Surface Management



**Your organization should **find and fix** your risks  
before your attackers can exploit them**

**Cortex Xpanse** is an active attack surface management solution that helps your organizations find & fix your known and unknown internet-connected risks.



# The Data



# The Internet is Small

5.27B

Webpages

4.4B

IPv4 addresses

$2^{128}$

Potential IPv6  
addresses



# What data do we have?



## Scanning Data

### Certificates



### Services



## Public Internet Data

### Registration



### Domain Resolutions



### Geo IP



### Cloud Ranges

173.245.48.0/20  
103.21.244.0/22



### Vulnerabilities and Exploits

CVE - 2019 - 1214

And  
More

Beam @ Xpanse, Palo Alto Networks

# 🔍 Beam @ Xpanse, Palo Alto Networks

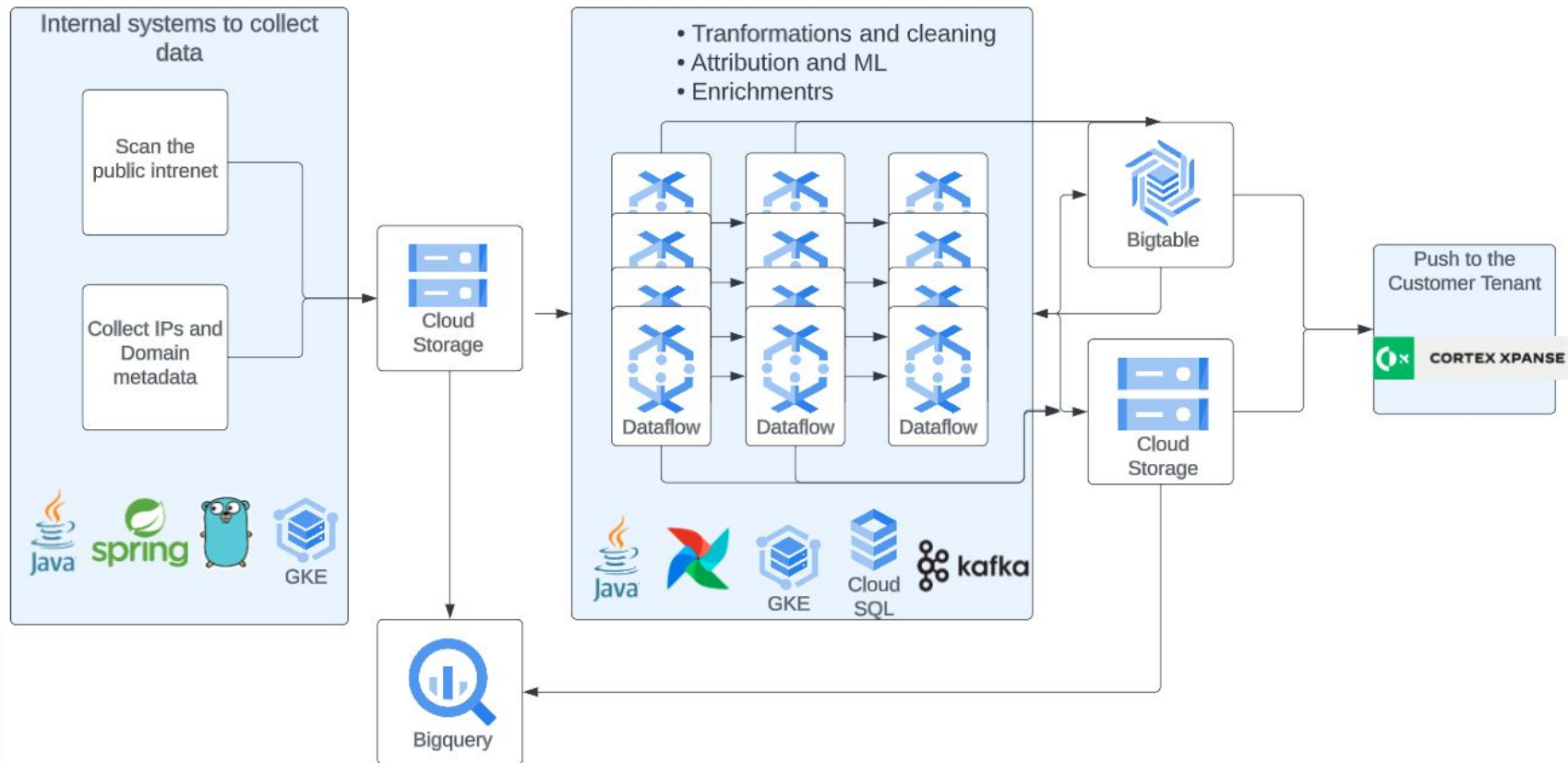


- We Process 10 Petabytes Daily with Apache Beam
- 200+ daily Pipelines
- Dataflow Runner, Java SDK
- Using Kubernetes (GKE) to run the Jobs
- If you're backend developer at Xpanse, you're a Beam developer





# Architecture





# Guidelines - PTransforms



- If you can, use Beam provided transforms
- Common Xpanse Beam library
  - Common company Transform operation
  - For example, all BigTable Writes and Reads are in the common library



The complete list of beam transforms:

<https://beam.apache.org/releases/javadoc/2.48.0/index.html?org/apache/beam/sdk/transforms/package-summary.html>



# Guidelines



- Test your pipeline!
  - Test every PTransform individually
  - Test your pipeline

```
public class WordCountTest {  
    // Our static input data, which will comprise the initial PCollection.  
    static final String[] WORDS_ARRAY = new String[] {  
        "hi there", "hi", "hi sue bob",  
        "hi sue", "", "bob hi"};  
  
    static final List<String> WORDS = Arrays.asList(WORDS_ARRAY);  
  
    // Our static output data, which is the expected data that the final PCollection must match.  
    static final String[] COUNTS_ARRAY = new String[] {  
        "hi: 5", "there: 1", "sue: 2", "bob: 2"};  
  
    // Example test that tests the pipeline's transforms.  
  
    public void testCountWords() throws Exception {  
        Pipeline p = TestPipeline.create();  
  
        // Create a PCollection from the WORDS static input data.  
        PCollection<String> input = p.apply(Create.of(WORDS));  
  
        // Run ALL the pipeline's transforms (in this case, the CountWords composite transform).  
        PCollection<String> output = input.apply(new CountWords());  
  
        // Assert that the output PCollection matches the COUNTS_ARRAY known static output data.  
        PAssert.that(output).containsInAnyOrder(COUNTS_ARRAY);  
  
        // Run the pipeline.  
        p.run();  
    }  
}
```

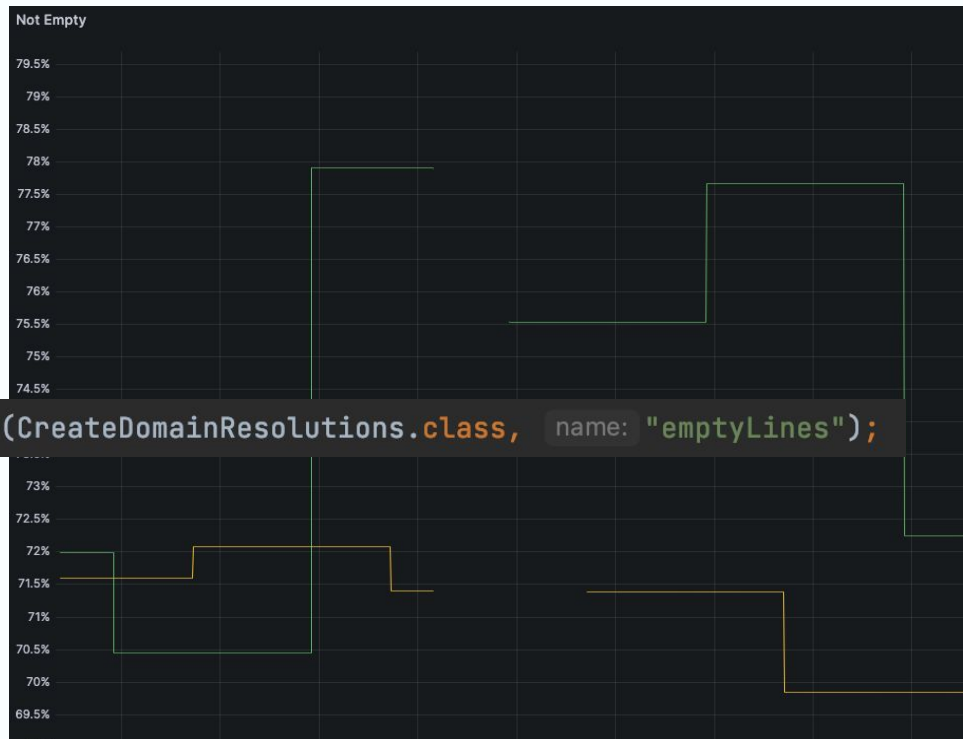


# Guidelines



- Monitor your pipeline!
  - Use Beam metrics
  - Track the jobs run time

```
private final Counter emptyLines = Metrics.counter(CreateDomainResolutions.class, name: "emptyLines");
```



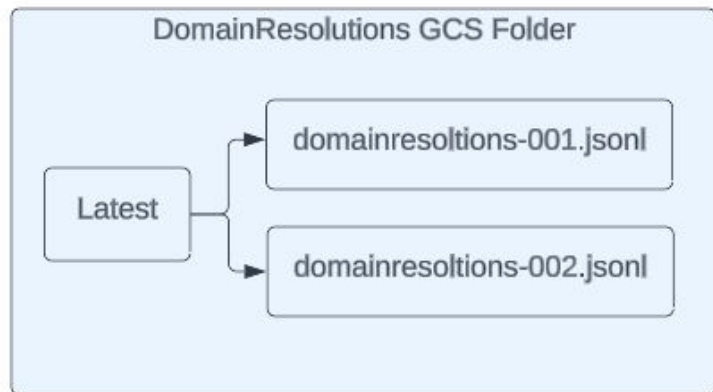




# Guidelines - GCS Write



- Write to Cloud Storage (GCS) a “latest file”
- The latest file content references to GCS files
- Batch pipelines can read the latest files easily



## Using our common writer transform:

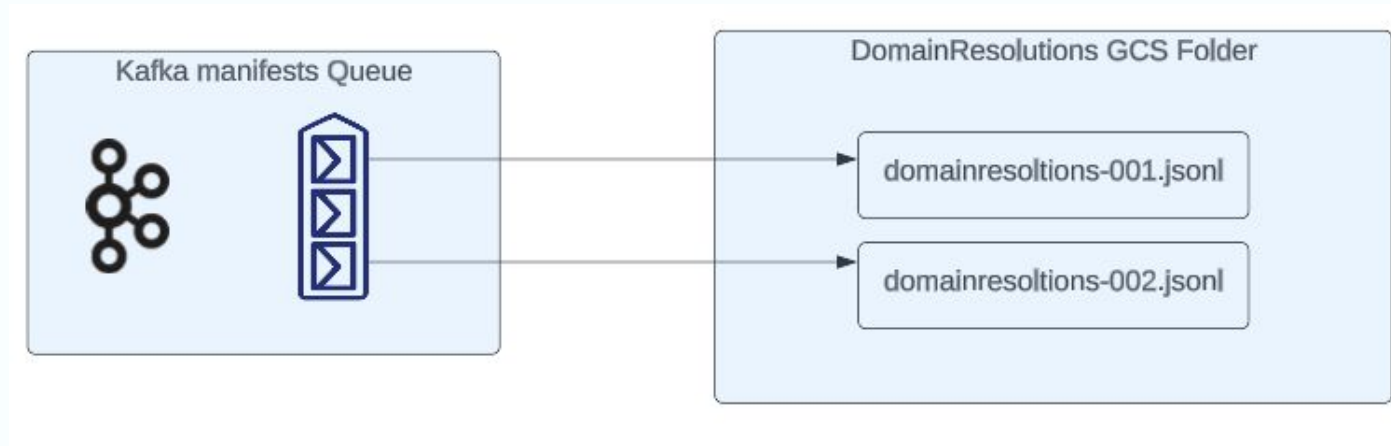
```
domainResolutions.apply(  
    name: "Write to Files with latest",  
    FileWriterWithLatest.builder()  
        .latest(true)  
        .suffix(".jsonl")  
        .latestPath(path + "/latest")  
        .outputPath(path)  
        .build());
```



# Guidelines - Kafka Manifests



- Use GCS as external storage to Kafka
- Send to kafka references the GCS files



# Guidelines Usage– A Performance Tuning Case



# The story of the DNS Pipeline



- Domain Resolutions data has a lot of garbage in it!
- We have a pipeline to aggregate similar subdomains and cleaning

```
ns1.mydomainname.com.  
ns2.mydomainname.com.  
mydomainname.com.  
www.mydomainname.com.  
mydomainname.com.  
www.mydomainname.com.  
mail.mydomainname.com.  
webmail.mydomainname.com.
```

```
A      194.23.253.196  
A      194.23.254.196  
A      194.23.253.196  
A      194.23.253.196  
AAAA   4001:41d0:2:80c4::  
AAAA   4001:41d0:2:80c4::  
A      194.23.253.196  
A      194.23.253.196
```



# The story of the DNS Pipeline



- The pipeline kept getting more records, increasing run time and cost
- 75% of the cost was due to shuffles!

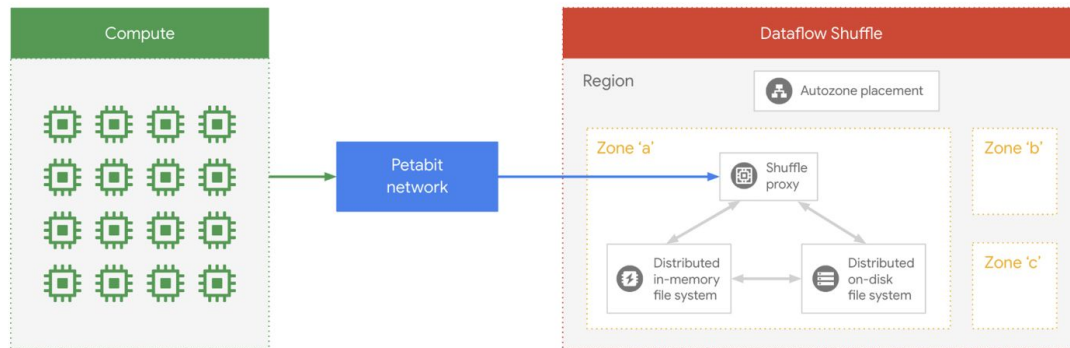




# Shuffles



- The data in Beam moves
- The dataflow shuffle service
  - Available for batch jobs
  - moves the shuffle operation out of the worker VMs and into the Dataflow service backend





# Beam Coders



- Java Object - > Byte array - > Java Object
- Beam has some efficient Coders implementations, for example:
  - ProtoCoder
  - AvroCoder
  - SchemaCoder
- The idea:
  - SerializableCoder (Default coder) → Custom Coder





# Custom Coder!



```
public class DomainResolution {
    private final String domainName;
    private final String ip;
}

public class DomainResolutionCoder extends Coder<DomainResolution> {
    private static final Coder<String> STRING_CODER = StringUtf8Coder.of();

    @Override
    public void encode(final DomainResolution value, final OutputStream outputStream)
        throws IOException {
        STRING_CODER.encode(value.getDomainName(), outputStream);
        STRING_CODER.encode(value.getIp(), outputStream);
    }

    @Override
    public DomainResolution decode(final InputStream inputStream) throws IOException {
        return DomainResolution.builder()
            .domainName(STRING_CODER.decode(inputStream))
            .ip(STRING_CODER.decode(inputStream))
            .build();
    }
}
```



# Q The Results



- ~50% cost improvement In shuffle!
- Tens of thousands of dollars saved yearly



## Before

Total Shuffle data processed ?	14.54 TB
Billable Shuffle data processed ?	11.98 TB

## After

Total Shuffle data processed ?	8.7 TB
Billable Shuffle data processed ?	6.14 TB



# To Summarize the Process



- We recognized a scaling issue using our **monitoring infrastructure**
- Developed a reusable solution, exposed in our **common library**
- **Tested** the new solution
- Deployed and tracked it using our monitoring infrastructure

Lior Dadosh

# QUESTIONS?