



elastic

GOVTECH
SINGAPORE



StackOps Workshop: 101

Kick start your observability journey today

November 2024

StackOps x Elastic

Agenda

Introduction and Overview

- Introduction to StackOps
- Why Observability

Planning

- Sizing deployment
- Sign up & Provision

Getting the data in / data ingestion

- VPCe Configuration, Architecture, benefits, security and cost
- Common AWS Collection Patterns and Comparison (EC2, Cloudtrail, S3/SNS/SQS)
- StackOps Observability Accelerator (SOA)

Hands-on Intro

- Lab UI & Elastic Cloud UI Walkthrough

Lab 1 & 2: Add Elastic Agent & Investigating System Logins

- How to Ingest Logs and Metrics
- Intro to Kibana UI for Log search, metrics, host views, default dashboards
- Clone a dashboard, Customisation of Dashboards, Alerts, Thresholding, Dashboard proficiency

OpenTelemetry, ECS, StackOps extended schema

- Data Dictionary, Data Profiling Dashboard benefits

Lab 3: Analyzing a Spike in Log Data

- Log Rate Analysis and Log Pattern Analysis
- Introduce Anomaly Detection

Lab 4: SLO/SLI Implementation

- Labs

Wrap Up

- Docs portal, Telegram, Chatbot

Telegram



<https://go.gov.sg/stackops-community>

Q & A

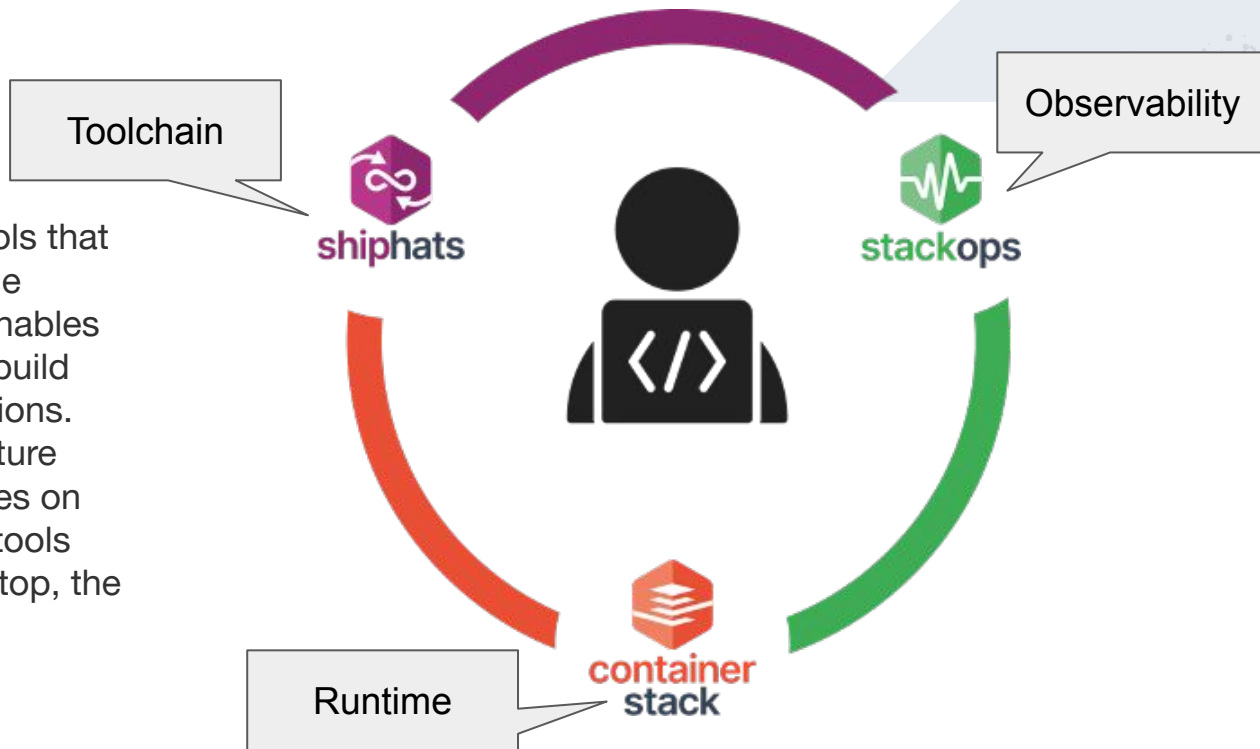


<https://pigeonhole.at/STACKOPS101>

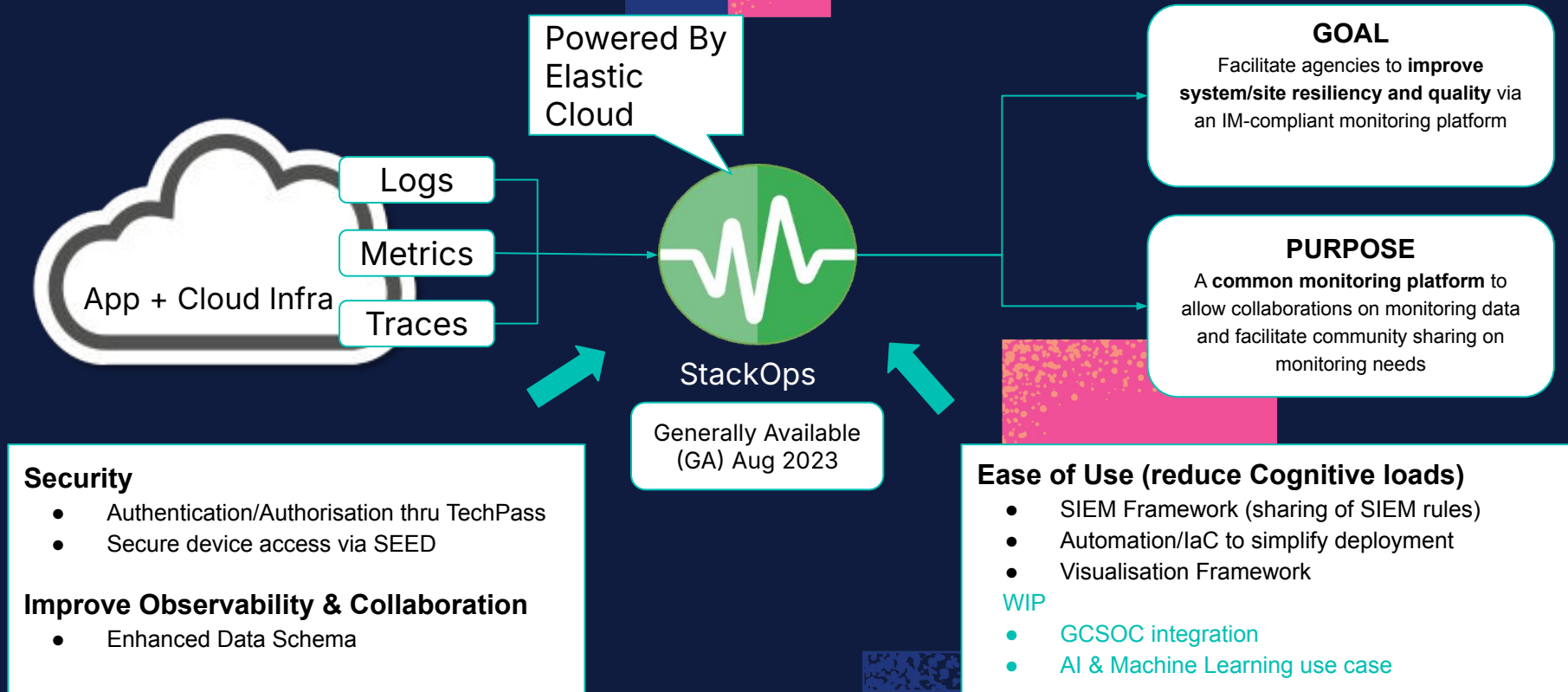
Introduction and Overview

SG Tech Stack (SGTS) Base Layer

SGTS is a set of platform tools that streamlines and simplifies the development process and enables code reuse across WOG to build secure, high quality applications. SGTS has a 2-layer architecture where the Base Layer focuses on standardising development tools and environment and at the top, the Services Layer drives code reusability within WOG.



StackOps - SGTs Observability Solution



Why Observability ?

- Outside-in approach Citizens and Businesses centricity





https://en.wikipedia.org/wiki/OODA_loop

Define: Observability

Observability pertains to how a system's internal state can be understood by examining its external outputs, especially its data.

In the context of modern application development, observability refers to the collection and analysis of data — logs, metrics and traces — from a wide variety of sources, to provide detailed insight into the behavior of applications running in your environments. It can be applied to any system that you build and want to monitor.

Observability is crucial for today's dynamic architectures and multi-cloud computing environments. It allows software engineers, IT, DevOps, and Site Reliability Engineering (SRE) teams to decode telemetry data. This is done with the help of visualizations — dashboards, service dependency maps, and distributed traces — as well as AIOps and machine learning approaches. With the right observability solution, you can understand how your applications, services, and infrastructure are performing to track and respond to issues.

Why: Observability

Observability is important because it allows teams to evaluate, monitor, and improve the performance of distributed IT systems. It is far more effective than traditional monitoring methods. An end-to-end observability platform can help breach silos and foster collaboration. Issues can be proactively diagnosed, analyzed, and traced back to their origins.

Observability gives IT organizations true visibility across the many data sources in cloud-native environments. The three pillars of observability are logs, metrics, and traces. Full-stack observability allows you to track the performance of your multi-cloud ecosystem in real-time and historically. You can synthesize data from endpoints and services generated by hardware, software, cloud infrastructure components, containers, open-source, microservices, and more.

Observability helps organizations:

- Discover and analyze the significance of performance incidents to their business
- Increase the efficiency of software development life cycles
- Accelerate problem resolution and root cause analysis
- Improve end-user experiences
- Fortify application security

Google SRE - Four Golden Signals

01	Latency	• The time it takes to service a request • important to distinguish between the latency of successful requests and the latency of failed requests
02	Traffic	• Measure of how much demand is being placed on your system • For a web service, this measurement is usually HTTP requests per second
03	Errors	• The rate of requests that fail, either explicitly (e.g., HTTP 500s) • Implicitly (for example, an HTTP 200 success response, but coupled with the wrong content)
04	Saturation	• How "full" your service is. • A measure of your system fraction, emphasizing the resources that are most constrained • e.g., in a memory-constrained system, show memory; in an I/O-constrained system, show I/O

Planning and Sizing

Sizing Estimation

Part One, Storage sizing:

- Cloud Elements Calc
 - Input: Metrics/Logs
 - Output: Storage Requirements

Part Two, Estimate Cloud Costs

- Elastic Cloud Calc
 - Input: Storage Requirements
 - Output: Cloud Costs

Part One: Data Source Sizing

Refer to Cloud Elements Calc

Integration - Metrics		Qty	Collection Frequency	Sample Size (Bytes)	Retention in Days			No of Collections	Daily Storage (MB)	Raw Storage (GB)			Total Stored (GB)
					Hot	Warm	Frozen			Hot	Warm	Frozen	
API Gateway	2		Every minute	2,272.00	3	27	336	1,440.00	6.24	0.02	0.16	2.06	2.23
Billing	1		Every Day	1,456.00	3	27	336	1.00	0.00	0.00	0.00	0.00	0.00
CloudWatch	50		Every 5 minutes	1,262.00	3	27	336	288.00	17.33	0.05	0.46	5.69	6.19
DynamoDB	2		Every 5 minutes	2,100.00	3	27	336	288.00	1.15	0.00	0.03	0.38	0.41
EBS	50		Every 5 minutes	2,406.00	3	27	336	288.00	33.04	0.10	0.87	10.84	11.81
EC2	50		Every 5 minutes	4,746.00	3	27	336	288.00	65.18	0.19	1.72	21.39	23.30
EC3	2		Every 5 minutes	2,142.00	3	27	336	288.00	1.18	0.00	0.03	0.39	0.42
ELB	8		Every minute	2,346.00	3	27	336	1,440.00	25.77	0.06	0.68	8.49	9.21
EMR	2		Every 30 minutes	1,438.00	3	27	336	48.00	0.13	0.00	0.00	0.04	0.05
Fargate	4		Every 5 minutes	5,146.00	3	27	336	288.00	5.65	0.02	0.15	1.86	2.02
Kinesis	2		Every 5 minutes	1,892.00	3	27	336	288.00	1.04	0.00	0.03	0.34	0.37
Network Firewall	4		Every minute	1,608.00	3	27	336	1,440.00	8.83	0.03	0.23	2.90	3.16
Lambda	2		Every 5 minutes	1,771.00	3	27	336	288.00	0.97	0.00	0.03	0.32	0.35
NAT Gateway	4		Every minute	3,078.00	3	27	336	1,440.00	16.91	0.05	0.45	5.55	6.04
Route53	1		Every 30 minutes	3,543.00	3	27	336	48.00	0.16	0.00	0.00	0.05	0.06
RDS	2		Every 5 minutes	5,438.00	3	27	336	288.00	2.59	0.01	0.08	0.98	1.07
S3 Daily Storage	1		Every Day	1,557.00	3	27	336	1.00	0.00	0.00	0.00	0.00	0.00
S3 Request	1		Every 30 minutes	1,775.00	3	27	336	48.00	0.08	0.00	0.00	0.03	0.03
S3 Storage Lens	1		Every 30 minutes	3,896.00	3	27	336	48.00	0.18	0.00	0.00	0.06	0.06
SNS	2		Every 5 minutes	1,700.00	3	27	336	288.00	0.93	0.00	0.02	0.31	0.33
SQS	2		Every 5 minutes	1,878.00	3	27	336	288.00	1.03	0.00	0.03	0.34	0.37
Transit Gateway	2		Every minute	2,602.00	3	27	336	1,440.00	7.15	0.02	0.19	2.34	2.55
Usage	1		Every Day	1,491.00	3	27	336	1.00	0.00	0.00	0.00	0.00	0.00
VPN	1		Every minute	1,143.00	3	27	336	1,440.00	1.57	0.00	0.04	0.52	0.56
								Metrics Raw Total:	197.53	0.58	5.21	64.81	70.60
								Metrics Total:	278.86	0.82	7.35	38.89	99.67

Integration - Logs		Qty	Events Per Day	Sample Size (Bytes)	Retention in Days			Daily Events	Daily Storage (MB)	Raw Storage (GB)			Total Stored (GB)
					Hot	Warm	Frozen			Hot	Warm	Frozen	
API Gateway	2		5000	612.00	3	27	336	5,000.00	2.92	0.01	0.08	0.96	1.04
CloudFront	1		5000	3,546.00	3	27	336	5,000.00	16.91	0.05	0.45	5.55	6.04
CloudTrail	10		1000	2,841.00	3	27	336	10,000.00	270.94	0.79	7.14	88.90	96.84
CloudWatch	50		1000	781.00	3	27	336	50,000.00	1,862.05	5.46	49.10	610.88	665.54
EC2	50		1000	651.00	3	27	336	50,000.00	1,552.10	4.55	40.92	509.28	554.76
ELB	8		5000	2,883.00	3	27	336	40,000.00	879.82	2.58	23.20	288.69	314.47
EMR	8		5000	663.00	3	27	336	40,000.00	292.33	0.96	8.33	96.39	105.68
Guardduty	1		5000	11,266.00	3	27	336	5,000.00	53.72	0.16	1.42	17.63	19.20
Inspector	1		5000	10,269.00	3	27	336	5,000.00	48.97	0.14	1.29	16.07	17.50
Network Firewall	4		10000	4,040.00	3	27	336	40,000.00	616.46	1.81	16.25	202.27	220.33
Route 53 Public	1		5000	2,416.00	3	27	336	5,000.00	11.52	0.03	0.30	3.78	4.12
Route 53 Resolver	1		5000	3,816.00	3	27	336	5,000.00	18.20	0.05	0.48	5.97	6.50
S3 Request	1		2000	4,408.00	3	27	336	2,000.00	8.41	0.02	0.22	2.76	3.01
Security Hub Findings	1		10	21,963.00	3	27	336	10.00	0.21	0.00	0.07	0.07	0.07
Security Hub Insights	1		10	33,754.00	3	27	336	10.00	0.35	0.00	0.07	0.11	0.12
VPC Flow	2		100000	1,821.00	3	27	336	200,000.00	694.66	2.04	18.32	227.93	248.29
WAF	2		10000	2,908.00	3	27	336	20,000.00	110.93	0.32	2.92	36.40	39.65
								Logs Raw Total:	6,828.41	20.01	180.05	2,187.96	2,440.62
								Logs Total:	19,280.23	56.49	508.37	2,625.55	6,891.17

Select your services and input an estimation.

- Metrics by interval
- Logs by events per day

Part One: Storage Requirements

Refer to Cloud Elements Calc

Grand Raw Total:	Raw Storage (TB)				
	Daily Volume	Hot	Warm	Frozen	Total Stored (TB)
	6.86	0.02	0.18	2.20	2.45
Grand Total:	Indices Storage (TB)				
	Daily Volume	Hot	Warm	Frozen	Total Stored (TB)
	19.10	0.06	0.50	2.60	6.83
Caters for Index, Compression, Replica and Watermark Buffer					

Part Two: Estimate Elastic Cloud Cost

Use: <https://cloud.elastic.co/pricing>

Select cloud provider (AWS, Azure or GCP)

Select region (Singapore Region)

Select a hardware profile (See next slide)

Elastic Cloud (Elasticsearch Service) Pricing Calculator

Use the pricing calculator to get a reference price for a given configuration of Elastic. Depending on your use case, you choose the optimal [hardware profile](#). Each hardware profile is a unique blend of virtual storage, RAM, and vCPUs. Depending on the hardware profile, you adjust the size of the resources (RAM, vCPU and disk capacity) you dedicate to the constituents of the Stack. At the end of the day, Elastic Cloud simply charges against the total capacity consumed. You are also charged for data transfer and storage (DTS) costs.

Note that your exact optimal configuration, and consequent pricing, depends on your actual workload. If you're not sure, we recommend that you start with Elastic Cloud monthly billing and configure the resources to adequately support your workload to understand the associated cost. You can start by signing up for a [free trial](#). Or, you can [contact us](#) to come up with an optimal configuration, and associated price, for your use case.

1 Where do you need it deployed?

Cloud provider

 Amazon Web Services

Region

 Singapore (ap-southeast-1)

Hardware profile ⓘ

Storage optimized

Summary

Standard Gold Platinum **Enterprise**

[Compare subscriptions](#)

ELASTICSEARCH

Hot storage **30 GB**

Hot memory **1 GB**

Hourly rate **\$0.0532**

Choosing a hardware profile

Hardware Profiles affects the hot tier.

Choose Compute Optimised if your use cases requires more compute.

- E.g More concurrent alerts, anomaly detection jobs
- **Performance Efficiency:** Compute-optimized hardware profiles provide higher CPU-to-memory ratios, which are beneficial for compute-intensive workloads such as indexing and searching in Elasticsearch. This ensures that Elasticsearch operations are performed quickly and efficiently, leading to faster query responses and data processing.
- **Cost Optimization:** By selecting a hardware profile that is tailored to the compute needs of Elasticsearch, you can avoid over-provisioning resources. This means you only pay for the compute power you need, which can result in cost savings while still maintaining optimal performance for your Elasticsearch cluster.
- **Scalability and Flexibility:** Compute-optimized profiles are designed to handle variable workloads by providing the necessary compute resources to scale up during high-demand periods. This flexibility ensures that Elasticsearch can maintain stable performance even as the workload grows, without the need for constant manual adjustments to the hardware configuration.

Choosing a hardware profile

Hardware Profiles affects the hot tier.

Choose Storage Optimised (Dense) if your use cases requires more storage.

- E.g Longer data retention with minimal searches.
- **High Storage Capacity:** Storage-optimized dense hardware profiles offer a larger storage capacity relative to their compute (CPU and memory) resources. This is ideal for Elasticsearch workloads that require storing and processing large volumes of data, such as logging and time-series data, without the need for extensive computing power.
- **Cost-Effective Data Retention:** With more storage available at a lower cost, these profiles enable cost-effective data retention strategies. This is particularly beneficial for use cases where data needs to be retained for long periods for compliance, analysis, or historical reference.
- **I/O Optimized Performance:** Storage-optimized profiles are typically equipped with fast I/O capabilities to handle high disk I/O operations, which is crucial for Elasticsearch tasks like merging segments and performing bulk read/write operations. This ensures that Elasticsearch can manage large datasets efficiently without becoming I/O bound.

Configure Hot/Warm/Frozen: Example

Select the Size per zone and check that you have enough storage stated in Total (size x zone).

Grand Total:	Indices Storage (TB)				
	Daily Volume	Hot	Warm	Frozen	Total Stored (TB)
	19.10	0.06	0.50	2.60	6.83
Caters for Index, Compression, Replica and Watermark Buffer					

2

Size your deployment

 Elasticsearch

Hot data and Content tier [info](#)

Nodes in this tier ingest and process frequently queried data.

Size per zone

870 GB storage | 29 GB RAM | 4 vCPU

Availability zones

☐ 1 zone ☒ 2 zones ☐ 3 zones

Total (size x zone)

1.7 TB storage | 58 GB RAM | 8 vCPU

Match the remaining tiers.

Grand Total:	Indices Storage (TB)				
	Daily Volume	Hot	Warm	Frozen	Total Stored (TB)
	19.10	0.06	0.50	2.60	6.83
Caters for Index, Compression, Replica and Watermark Buffer					

 **Elasticsearch**

Hot data and Content tier [info](#)

Hot data and content tier is the default tier for Elasticsearch. It is designed for high performance and low latency. It is suitable for hot data and content that is frequently accessed.

Size per zone: 1.7 TB storage | 58 GB RAM | 8 vCPU

Availability zones: ☐ 1 zone ☒ 2 zones ☐ 3 zones

Total (size x zone): 1.7 TB storage | 58 GB RAM | 8 vCPU

Warm data tier [info](#)

Store less-frequently queried data on a warm tier for increased cost savings.

Size per zone: 5.57 TB storage | 30 GB RAM | 3.9 vCPU

Availability zones: ☐ 1 zone ☒ 2 zones ☐ 3 zones

Total (size x zone): 11.13 TB storage | 60 GB RAM | 7.8 vCPU

Cold data tier

Store rarely queried data on a cold tier for increased cost savings. [info](#)

[+ Add capacity](#)

Frozen data tier [info](#)

Maximize savings by archiving data on a frozen tier. Your storage size comes with a cached storage amount, which lets you keep the most frequently accessed data in memory.

Size per zone: 46.88 TB storage | 30 GB RAM | 3.9 vCPU

Includes 2.17 TB of cached storage for 46.88 TB of storage.

Availability zones: ☒ 1 zone ☐ 2 zones ☐ 3 zones

Total (size x zone): 46.88 TB storage (2.17 TB cached storage) | 30 GB RAM | 3.9 vCPU



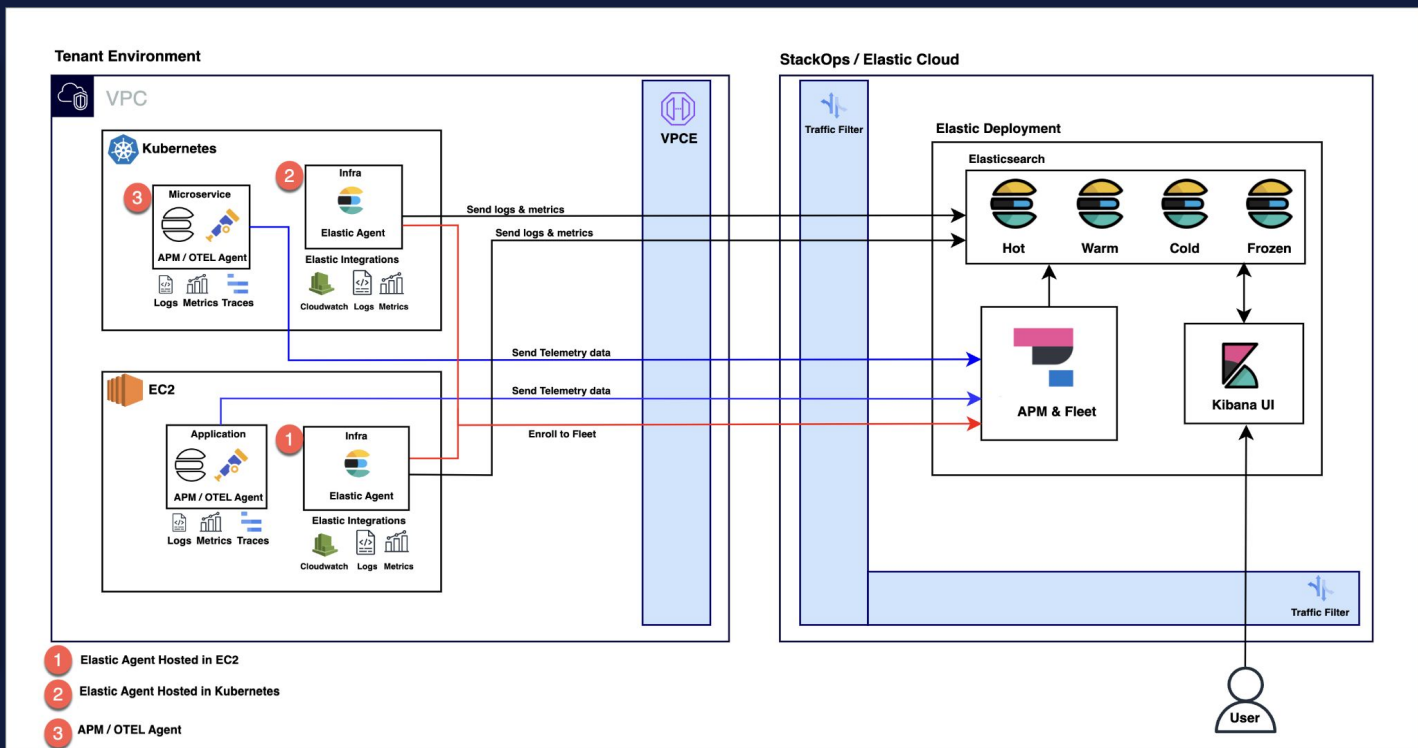
How do we get data in?



Secured Elastic With Traffic Filter

To open network connectivity.
Guidelines on
<https://docs.developer.tech.gov.sg/docs/stackops-documentation/create-endpoint>

Added Value:
Security and reduced cost



Ingest Decision Tree

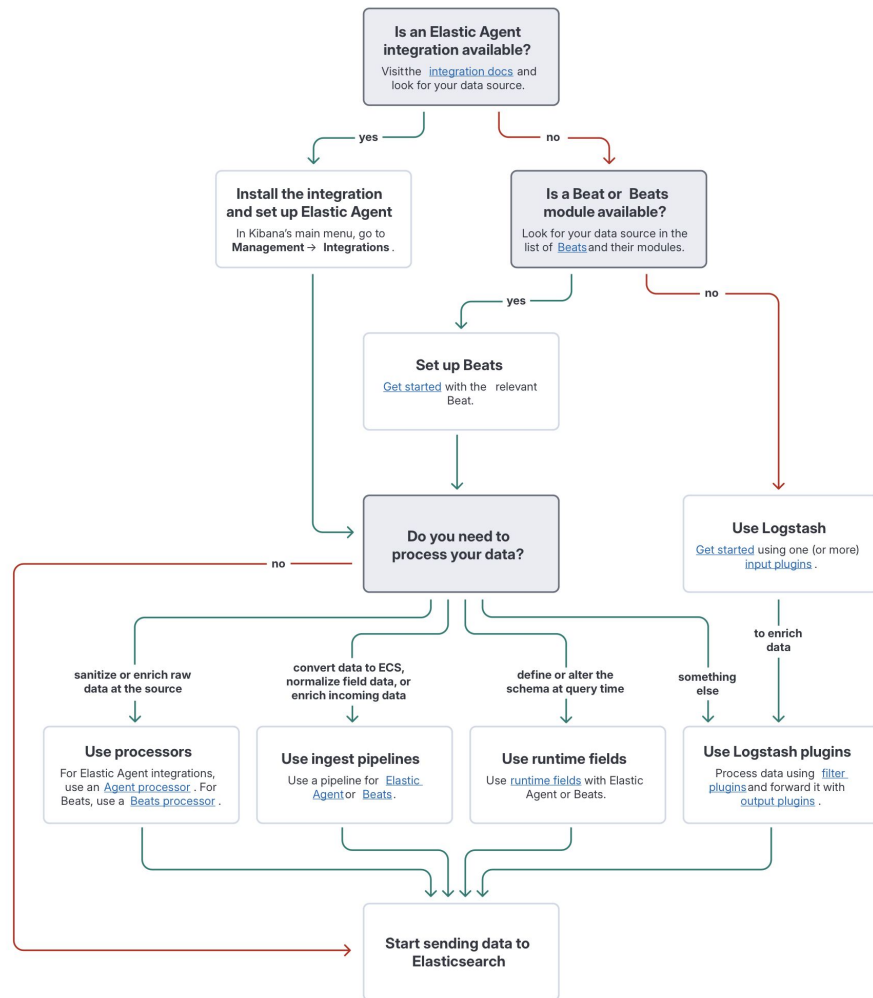
How to ingest data

<https://www.elastic.co/guide/en/cloud/current/ec-cloud-ingest-data.html>

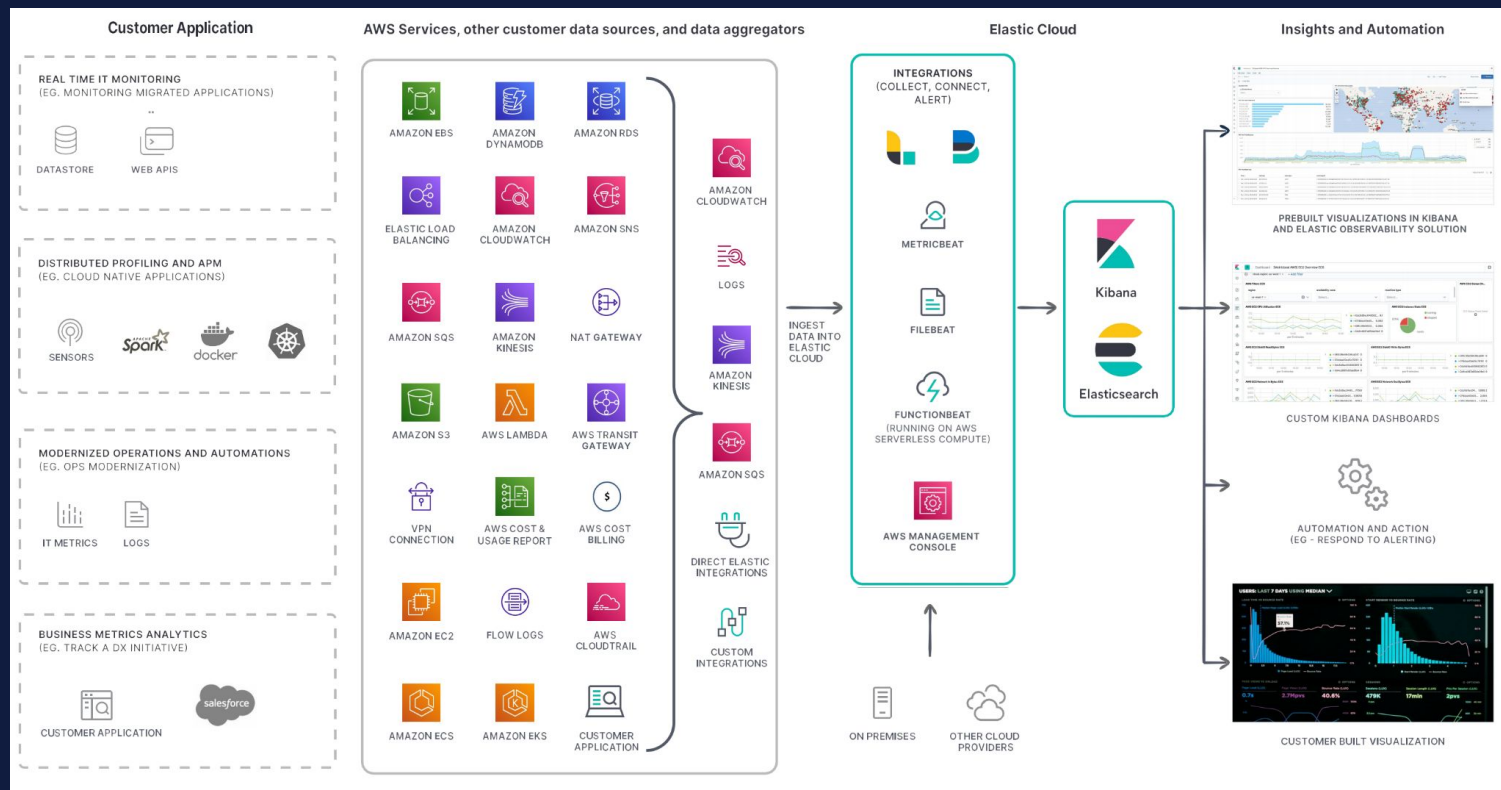
StackOps Recommendations:

<https://docs.developer.tech.gov.sg/docs/stackops-documentation/data-ingestion-methods>

Added Value: Standardised collection patterns



Elastic-AWS Integrations



StackOps Observability Accelerator (SOA)

StackOps tool to enable users and accelerate their observability implementations.



SIMPLIFY
CONFIGURATION
PROCESS



REDUCES SETUP TIME
FROM DAYS TO <1
HOUR

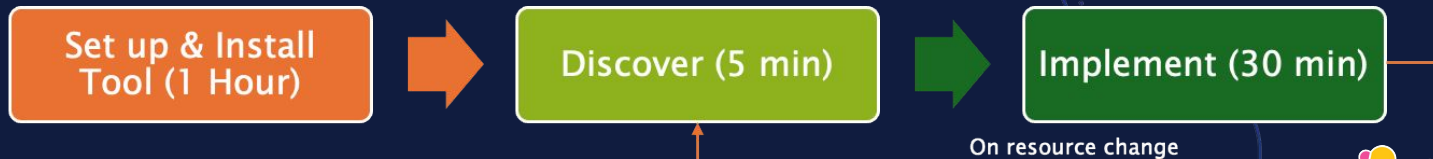


PERIODIC DISCOVERY
OF RESOURCE
CHANGES

Before SOA



After SOA



StackOps Observability Accelerator

StackOps tool to enable users and accelerate their observability implementations.



STACK 2024 – StackOps Observability Accelerator (SOA)

Hands-on Labs

Lab 1 - Ingest & Investigate Logs

<https://ela.st/stackops-101-lab>

Observability - Host Monitoring

Learn how to use Elastic Observability for host monitoring

Time limit: 1 hour

This track will teach you how to use Elastic Observability to monitor your hosts and applications. Setup Elastic Agent, monitor system logins and find the root cause of log spikes.

- 1 Add Elastic Agent
- 2 Investigating System Logins
- 3 Quiz - Logins
- 4 Add the Nginx Integration
- 5 View your Nginx Data
- 6 Analyzing a Spike in Log Data
- 7 Quiz - Log Spike
- 8 Investigating System Logins
- 9 Quiz - Log Spike

Start

In this short workshop you have access to the following modules:

- Host Monitoring
- Kubernetes Monitoring
- Application Performance Monitoring (APM)

Available content (3)



Observability - Host Monitoring

Learn how to use Elastic Observability for host monitoring

[View details](#)

Start

Don't go ahead to the quiz first

Choose your
Adventure

Elastic Menu

Elastic UI

Shell

Guide
Pin Left or Right

Check your
solution

The screenshot shows the Elastic Kibana Fleet management interface. The top navigation bar includes the 'elastic' logo, a search bar, and tabs for 'Fleet' and 'Agents'. The main content area is titled 'Add agent' and contains a form for adding new agents. The interface is annotated with several pink callout boxes:

- Elastic Menu:** Points to the top navigation bar.
- Elastic UI:** Points to the main content area.
- Shell:** Points to the 'Shell' tab in the top navigation bar.
- Choose your Adventure:** Points to the 'Add agent' button.
- Guide Pin Left or Right:** Points to the 'Add agent' button.
- Check your solution:** Points to the 'Check' button at the bottom right.

The 'Add agent' form includes a section titled 'What type of host are you adding?' with options for 'Elastic Agent on ECK' and 'Elastic Agent on ECS'. Below this, there is a table of agents with columns for 'Name', 'Agent policy', 'IPV4', 'Memory', 'Last activity', and 'Version'. The table shows two agents: 'elastic-agent-policy-1' and 'elastic-agent-policy-2'. The 'elastic-agent-policy-2' agent is highlighted with a red box and a red arrow pointing to the 'Add agent' button.

On the right side of the interface, there is a sidebar titled 'Elastic Agent Enrollment Guide' which provides a step-by-step guide for adding agents. The guide includes a progress bar at the top and a list of steps: '1. Initiate Agent Addition' and '2. Policy Creation'. The '1. Initiate Agent Addition' step includes a sub-step: 'Navigate to and click on 'Add Agent'.' The '2. Policy Creation' step includes a sub-step: 'Select Create new agent policy.' The sidebar also includes a table of agents and a 'Check' button at the bottom.

You can backtrack

Respond

Lab 1 Add Elastic Agent & Investigating System Logins

Remember [Y/n] for Agent installation

Clone the SSH Login Dashboards

Refer to Lab Guide for Kibana Exercises

- Edit the cloned dashboard
- Add a new panel
- Create log based errors alerts
- Create metrics threshold based alerts

Quiz Challenge

Collect

Normalize

Enrich

Detect

Investigate

Respond

Normalize data with ECS

Streamline analysis with a common schema

Analyzing a Spike in Log Data
Check the nginx logs in Discover

source.ip:10.184.4.2

Search *without* a common schema

```
src: 10.10.10.10  
OR client_ip: 10.10.10.10  
OR apache2.remote_ip: 10.10.10.10  
OR context.user.ip: 10.10.10.10  
OR src_ip: 10.10.10.10
```

Search *with* a common schema

```
source.ip: 10.184.4.2
```

Collect

Normalize

Enrich

Detect

Investigate

Respond

Enrich data

Contextualize data automatically and on demand

IP: 10.184.4.2

- `cloud.account.id: instruct-prod`
- `cloud.instance.name: p-pmfbdhuaflow0-kubernetes-vm`

**StackOps extended schema includes additional agency and project context:
System classification and Data sensitivity etc.**

Enrichment up front reduces standardising and reporting effort.

<https://docs.developer.tech.gov.sg/docs/stackops-documentation/create-ingest-pipelines>

Collect

Normalize

Enrich

Detect

Investigate

Respond

Lab 2 - Investigating Additional Log Spikes

<https://ela.st/stackops-101>

Observability - Host Monitoring

Learn how to use Elastic Observability for host monitoring

Time limit: **1 hour**

This track will teach you how to use Elastic Observability to monitor your hosts and applications. Setup Elastic Agent, monitor system logins and find the root cause of log spikes.

- 6 Add Elastic Agent
- 6 Investigating System Logins
- 6 Quiz - Logins
- 6 Add the Nginx Integration
- 6 View your Nginx Data
- 6 Analyzing a Spike in Log Data
- 6 Quiz - Log Spike
- 6 Investigating System Logins
- 6 Quiz - Log Spike

Start

In this short workshop you have access to the following modules:

- Host Monitoring
- Kubernetes Monitoring
- Application Performance Monitoring (APM)

Available content (3)



Observability - Host Monitoring

Learn how to use Elastic Observability for host monitoring

[View details](#)

Start

Lab 2 - Additional Nginx Logs

Machine Learning Aided Investigation

Collect metrics from Nginx instances

- Nginx port change 80 → 30001

AI Ops requires a Machine Learning node and it's not provisioned by default.

Collect

Normalize

Enrich

Detect

Investigate

Respond

Why ECS and structuring logs matters?

Why is there a spike on 12th June?

AI Ops Labs

- Log Rate Analysis

Sort the fields, click on a field name, observe the chart changes

Check your time range, 11th - 13th June 2023

Collect

Normalize

Enrich

Detect

Investigate

Respond

Why ECS and structuring logs matters?

Leverage on Machine Learning

What if we do not have a common schema?

Analysis type: Log rate spike

The median log rate in the selected deviation time range is higher than the baseline. Therefore, the analysis results table shows statistically significant items within the deviation time range that are contributors to the spike. The "doc count" column refers to the amount of documents in the deviation time range.

Field name	Field value	Log rate [Ⓢ]	Doc count	p-value [Ⓢ] ↑	Impact [Ⓢ] ↑	Actions
 source.address	103.136.43.123		1010	2.23e-308	High	...
 user_agent.original	Opera/10.46 (X11; Linux i686; en-US) Presto/2.8.339 Version/13.00		1010	2.23e-308	High	...
 source.as.organization.name	Ip Server LLC		1010	2.23e-308	High	...
 nginx.access.remote_ip_list	103.136.43.123		1010	2.23e-308	High	...
 source.ip	103.136.43.123		1010	2.23e-308	High	...
 related.ip	103.136.43.123		1010	2.23e-308	High	...
 url.path	/ddos-url-path		1010	2.23e-308	High	...
 url.original	/ddos-url-path		1010	2.23e-308	High	...
 user_agent.version	10.46		1012	2.23e-308	High	...

Collect

Normalize

Enrich

Detect

Investigate

Respond

What is OpenTelemetry?

OpenTelemetry is:

- An Observability framework and toolkit designed to create and manage telemetry data such as traces, metrics, and logs.
- Vendor- and tool-agnostic, meaning that it can be used with a broad variety of Observability backends, including open source tools like Jaeger and Prometheus, as well as commercial offerings.
- Not an observability backend like Jaeger, Prometheus, or other commercial vendors.
- Focused on the generation, collection, management, and export of telemetry. A major goal of OpenTelemetry is that you can easily instrument your applications or systems, no matter their language, infrastructure, or runtime environment. The storage and visualization of telemetry is intentionally left to other tools.

Why Now?

Language	Traces	Metrics	Logs
C++	Stable	Stable	Stable
C#/.NET	Stable	Stable	Stable
Erlang/Elixir	Stable	Development	Development
Go	Stable	Stable	Beta
Java	Stable	Stable	Stable
JavaScript	Stable	Stable	Development
PHP	Stable	Stable	Stable
Python	Stable	Stable	Development
Ruby	Stable	Development	Development
Rust	Beta	Alpha	Alpha
Swift	Stable	Development	Development

Why OpenTelemetry?

With the rise of cloud computing, microservices architectures, and increasingly complex business requirements, the need for software and infrastructure observability is greater than ever.

OpenTelemetry satisfies the need for observability while following two key principles:

- You own the data that you generate. There's no vendor lock-in.
- You only have to learn a single set of APIs and conventions.

Both principles combined grant teams and organizations the flexibility they need in today's modern computing world.

If you want to learn more, take a look at OpenTelemetry's [mission, vision, and values](#).

Main OpenTelemetry Components

OpenTelemetry consists of the following major components:

- A specification for all components
- A standard protocol that defines the shape of telemetry data
- Semantic conventions that define a standard naming scheme for common telemetry data types
- APIs that define how to generate telemetry data
- Language SDKs that implement the specification, APIs, and export of telemetry data
- A library ecosystem that implements instrumentation for common libraries and frameworks
- Automatic instrumentation components that generate telemetry data without requiring code changes
- The OpenTelemetry Collector, a proxy that receives, processes, and exports telemetry data
- Various other tools, such as the OpenTelemetry Operator for Kubernetes, OpenTelemetry Helm Charts, and community assets for FaaS

OpenTelemetry is used by a wide variety of libraries, services and apps that have OpenTelemetry integrated to provide observability by default.

OpenTelemetry is supported by numerous vendors, many of whom provide commercial support for OpenTelemetry and contribute to the project directly.

Observations About Observability



Data collection becomes commoditized via OpenTelemetry

- More services **expose OTel directly** or contribute Integrations to Collector
- APM Vendors begin to **drop proprietary APM agents and build upon OTel**

Observations About Observability

Vendor Differentiation Comes Through:

- Multi-signal analysis in unified User Experience
- ML-based single and multi-signal analytics for RCA
- Simple tools (BI for all!) for end users to analyze data
- Flexible deployment patterns (on-prem, Cloud, hybrid)
- Intelligent cost controls
- Workflow management (SLO, case management)
- Crowd-sourced (stackoverflow), LLM/chat-based AI Assistance

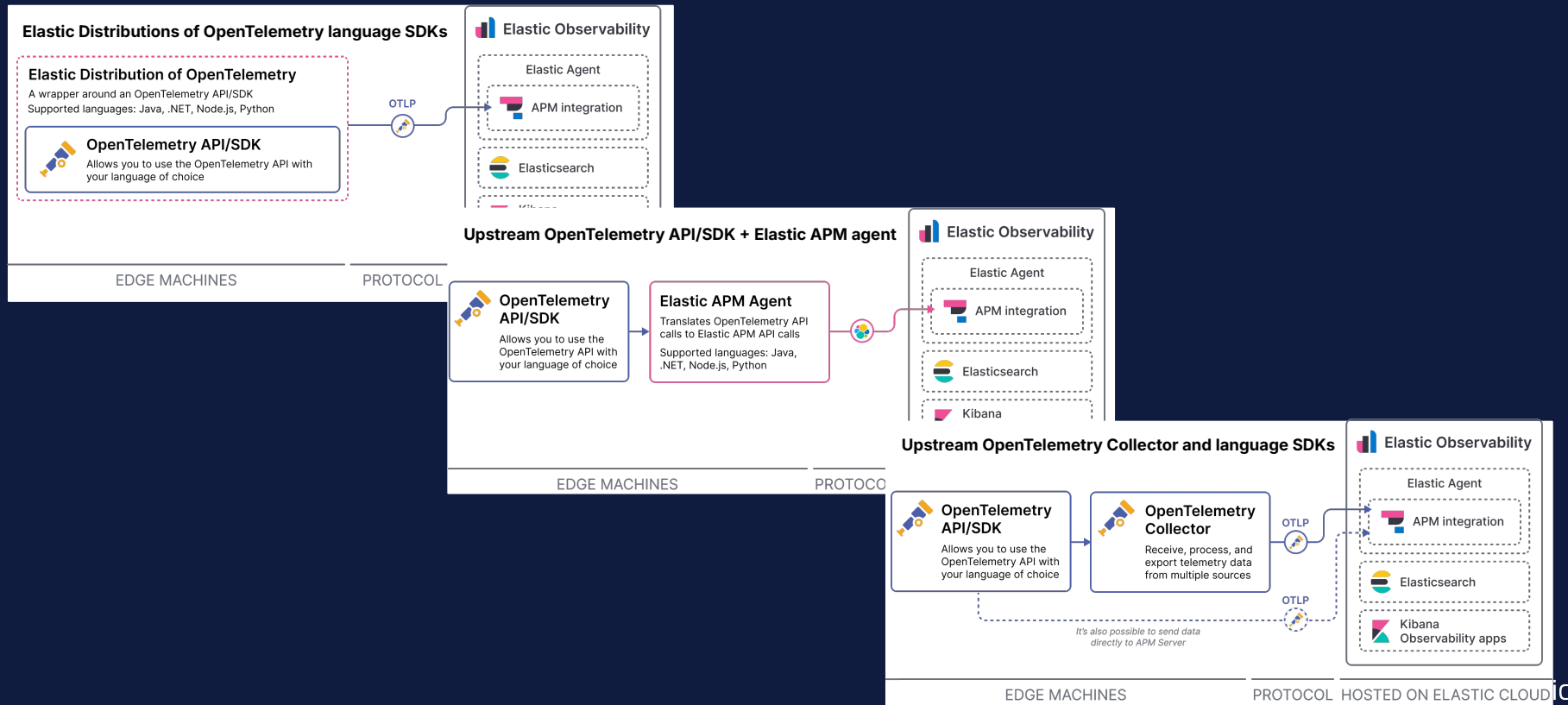
We are standardizing and investing on OpenTelemetry (OTel)

Make **OTel** the best data collection and processing framework

Base Elastic's data collection architecture for Observability and Security in **OTel**

Contribute to the success of **OTel** and help shape its future vs reacting

Elastic x OpenTelemetry



OpenTelemetry Distro

List of open source OpenTelemetry distributions maintained by third parties.

OpenTelemetry distributions are a way of customizing OpenTelemetry components so that they're easier to deploy and use with specific observability backends.

Any third-party can customize OpenTelemetry components with backend, vendor, or end-user specific changes. You don't have to use a distributions in order to use OpenTelemetry components, though distributions might facilitate usage under certain circumstances, such as specific vendor requirements.

<https://opentelemetry.io/ecosystem/distributions/>

<https://www.elastic.co/observability-labs/blog/tag/opentelemetry>

<https://www.elastic.co/observability-labs/blog/elastic-opentelemetry-sdk-distributions>

<https://www.elastic.co/observability-labs/blog/opentelemetry-demo-with-the-elastic-distributions-of-opentelemetry>

Lab 3: Analysing a Spike in Log Data

Lab 3 - Investigating Additional Log Spikes

<https://ela.st/stackops-101>

Observability - Host Monitoring

Learn how to use Elastic Observability for host monitoring

Time limit: **1 hour**

This track will teach you how to use Elastic Observability to monitor your hosts and applications. Setup Elastic Agent, monitor system logins and find the root cause of log spikes.

- 6 Add Elastic Agent
- 6 Investigating System Logins
- 6 Quiz - Logins
- 6 Add the Nginx Integration
- 6 View your Nginx Data
- 6 Analyzing a Spike in Log Data
- 6 Quiz - Log Spike
- 6 Investigating System Logins
- 6 Quiz - Log Spike

Start

In this short workshop you have access to the following modules:

- Host Monitoring
- Kubernetes Monitoring
- Application Performance Monitoring (APM)

Available content (3)



Observability - Host Monitoring

Learn how to use Elastic Observability for host monitoring

[View details](#)

Start

Log Rate / Pattern Analysis

Leverage on Machine Learning



Collect

Normalize

Enrich

Detect

Investigate

Respond

AI & Analytics: Correlate any type of data

Accelerate problem resolution and improve predictability



Anomaly detection & correlations



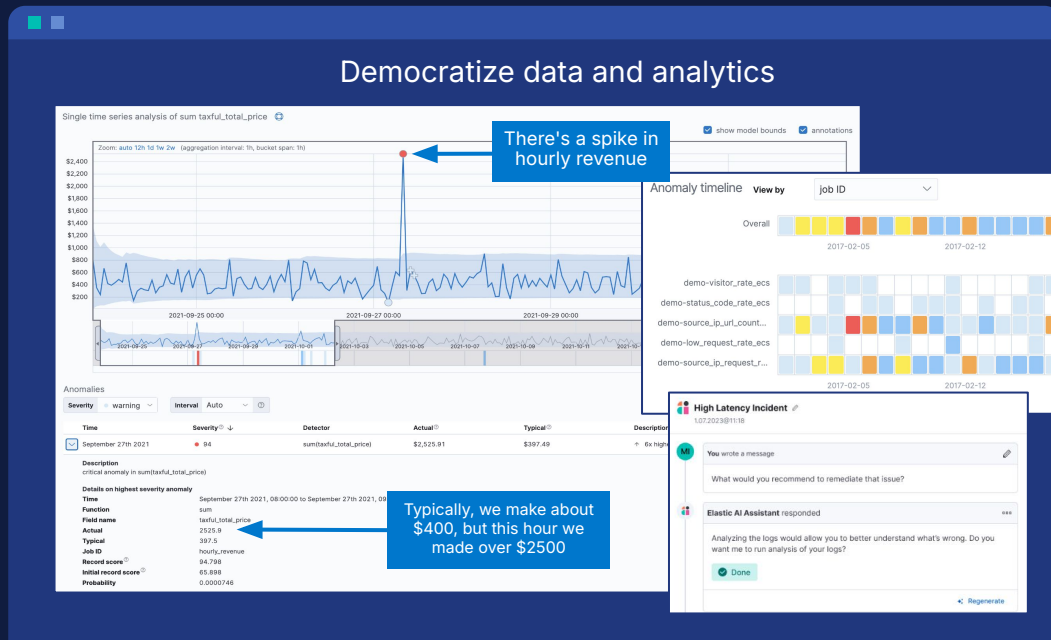
Log categorization



100+ out-of-the-box customizable ML models

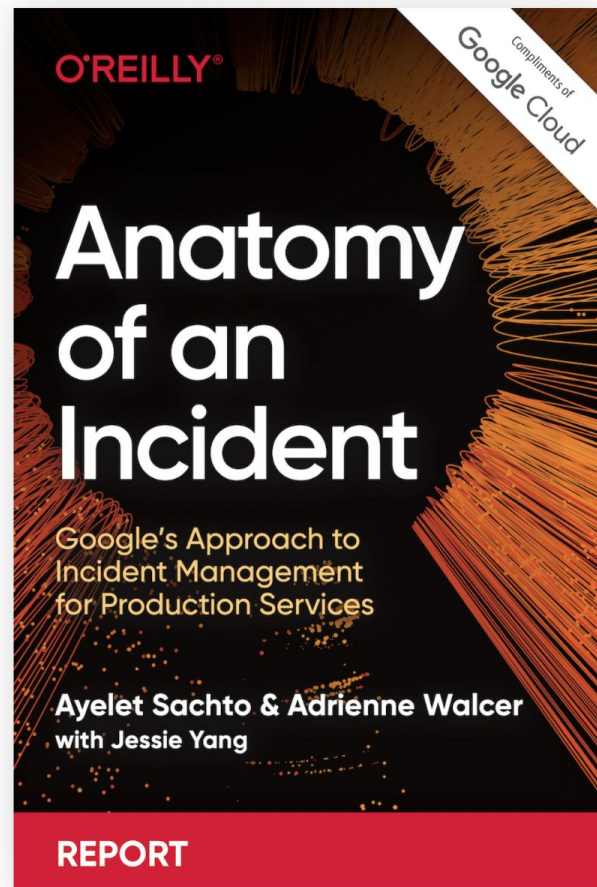


Generative AI powered by ESRE



Service Level Objectives (SLO)

Anatomy of an Incident - Google - Site Reliability Engineering



When it comes to system design, failure is inevitable. Scientists and engineers implement solutions based on the available information, without a complete knowledge of the future. You can't always anticipate the next zero-day event, viral media trend, weather disaster, or shift in technology. But you can be prepared to respond when incidents like these affect your systems.

With this report, SRE and DevOps practitioners, IT managers, and engineering leaders will explore methods to help your organization prepare for, respond to, and recover from incidents. With advice from Ayelet Sachto, Adrienne Walcer, and Jessie Yang, you'll learn how to be prepared to handle failure if and when it happens.

Learn the stages of the incident management lifecycle: preparedness, response, recovery, and mitigation

- Deal proactively with incidents: issues that escalate beyond metrics and alerts
- Be prepared: practice disaster role playing and incident response exercises
- Learn the characteristics of the incident-response organizational structure
- Examine steps to recovery and mitigation after an incident has occurred
- Conduct postmortems to analyze what went wrong
- Explore a real-world example from Google: The Mayan Apocalypse
- Learn how to measure and reduce incidents impact
- Use postmortems as a tool for prevention and psychological safety

[PDF](#)[EPUB](#)[MOBI](#)

<https://sre.google/resources/practices-and-processes/anatomy-of-an-incident/>

Alert Fatigue ? Or Manage by SLO

Use effective alerts to avoid alert fatigue. Use pages when you need immediate action. Only the right responders—the specific team and owners—should get the alerts.

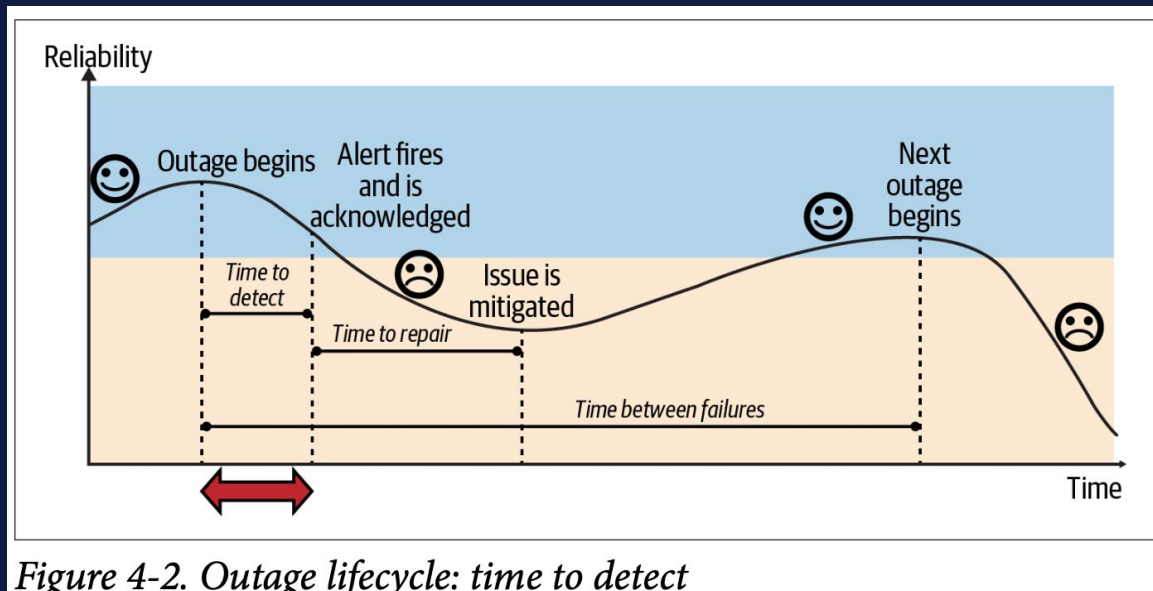


Figure 4-2. Outage lifecycle: time to detect

What matters to your users?

Service Level Indicator

The proportion of successful requests, as measured from the load balancer metrics.

Any HTTP status other than 500–599 is considered successful.

count of "api" http_requests which
do not have a 5XX status code
divided by
count of all "api" http_requests

<https://sre.google/workbook/slo-document/>

Create new SLO

1 Define SLI

Choose the SLI type

Custom Query

Index

Select a Data view

Timestamp field

Select a timestamp field

Query filter

Optional

Custom filter to apply on the index

Good query

Define the good events

Total query

Optional

Define the total events

Group by

Optional

Select an optional field to group by

QR Check In – Sports/Entertainment Venue

SLO: 1 failed out of 1000 successful check-ins = 99.9% successful check-ins

Error Budget: In this case: $100\% - 99.9\% = 0.1\%$

The error budget is the allowed number of errors or failures within a given time period. It's calculated as $100\% - \text{SLO}\%$. This means you have a 0.1% error budget. For every 1000 check-ins, you can tolerate 1 failure without violating your SLO.

Error Budget Calculation (using Occurrences method): $100,000 * 0.1\% = 100$ failures

Over a 30-day period, and you have 100,000 total check-ins during this time. You can have up to 100 failed check-ins over 30 days before you violate your SLO.

QR Check In – Sports/Entertainment Venue

Burn Rate:

The burn rate is how quickly you're using up your error budget compared to the rate you've allotted. A burn rate of 1 means you're using your error budget at exactly the expected rate. Higher than 1 means you're burning through your error budget too quickly.

Burn Rate = (Errors occurring / Time period) / (Total error budget / Total time)

For example, if you've had 20 failures in the first 3 days:

Burn Rate = (20 / 3) / (100 / 30) = 6.67

This burn rate of 6.67 means you're using your error budget 6.67 times faster than you should be.



Resume 11:50

Re-cap



Start your first StackOps workload

Start small and scale up

Resource based, StackOps scales proportionally to the following parameters:

- Daily ingest data volume ~ compute/storage
- Retention of data ~ storage
- Search Performance ~ compute

<https://docs.developer.tech.gov.sg/docs/stackops-documentation/get-started>

Key Takeaways

Importance of Observability

Key Takeaway:

Comprehensive visibility is crucial for maintaining reliable government services.

StackOps Capability:

StackOps facilitates comprehensive monitoring, leading to improved Mean Time to Resolution (MTTR) and operational efficiency.



Benefits:

- Enhance system visibility and troubleshooting capabilities
- Reduce Mean Time to Resolution (MTTR)
- Gain actionable insights for improved developer productivity

Establishing SLOs

Key Takeaway:

Clear performance targets ensure high-quality public services.

StackOps Capability:

StackOps provides tools for defining and tracking Service Level Objectives (SLOs) across government platforms.



Benefits:

- Set and maintain clear performance expectations for citizen-facing services
- Balance reliability and innovation in public sector tech
- Drive continuous improvement based on user experience metrics

OODA Loop in SRE

Key Takeaway:

The Observe Orient Decide Act loop enhances decision-making and response in dynamic government tech environments.

StackOps Capability:

StackOps supports each phase of the OODA loop with real-time data, analysis tools, and automated response capabilities.



Benefits:

- Observe: Continuously monitor system health and performance
- Orient: Quickly analyze and contextualize incidents within government services
- Decide: Make informed decisions based on data-driven insights
- Act: Implement rapid, effective responses to maintain service quality

Collaborative Problem-Solving

Key Takeaway:

Cross-agency collaboration enhances overall government service reliability.

StackOps Capability:

StackOps supports Whole-of-Government collaboration through shared monitoring and insights.



Benefits:

- Foster effective teamwork in cross-agency SRE teams
- Leverage shared knowledge and best practices across the public sector
- Improve overall government tech stack resilience through collective efforts

Q & A

<https://pigeonhole.at/STACKOPS101>



Call to Actions

Contact Us	enquiries_enp@tech.gov.sg
StackOps Docs	https://go.gov.sg/stackops-docs/
Future Workshops	https://go.gov.sg/stackops-learning-events
Telegram Community	Scan the code https://go.gov.sg/stackops-community
Learning Events	https://go.gov.sg/stackops-learning-events



Future Workshops

Level	Stage	Topics Covered
102	APM Monitoring <u>Sign up</u>	APM tracing Synthetic Monitoring SLO/SLI RUM Universal Profiling
103	SIEM	Creation of Rules Alerts integration Exception Handlings
201	Automation, Integration & Customising	Log/ingest Anomaly Detection (ML) Integration ML based Rules

We want to hear from you!

Take this 2 min survey

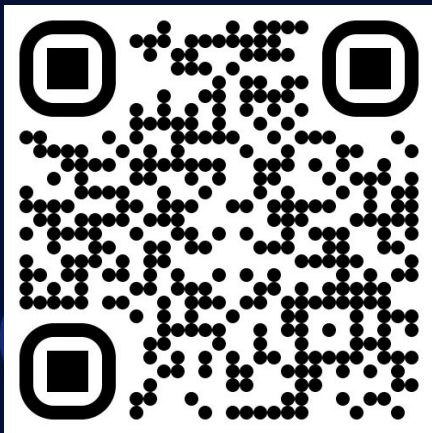
<https://go.gov.sg/stackops-workshop-feedback>





4th March 2025

Registrations are now open!
Secure your spots today!



Supercharge



Thank You