

Root Cause Analysis with Complex Event Processing

Kaspars Mednis

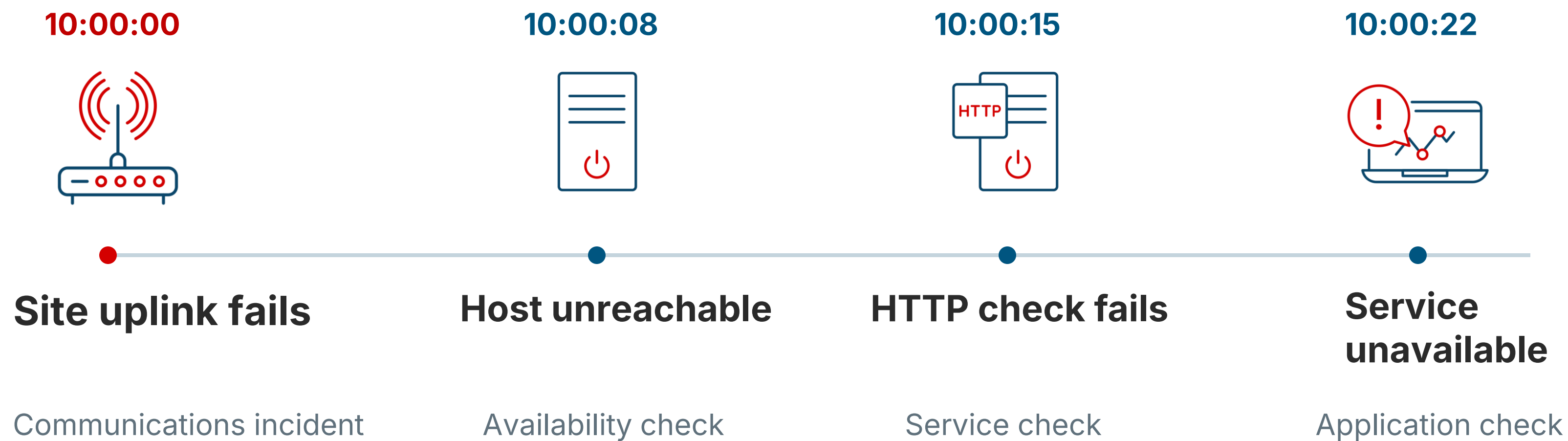
Lead Zabbix Trainer - Engineer

The root cause problem

One incident can produce many related alerts.

One failure, several alerts

A network outage can appear as several separate monitoring problems.



Which event explains the others?

Existing Zabbix approaches

Different mechanisms address different parts of the incident.

Trigger dependencies

Prevent dependent problem generation when a known upstream trigger is in the PROBLEM state".

Event correlation

!

Global event correlation is deprecated and may be removed in next releases.

Cause and symptoms

None

Simple (multiple)

Simple (once)

Cause and symptoms grouping

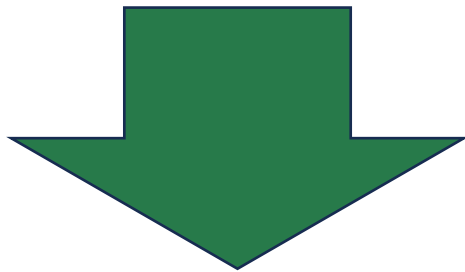
Pattern match

CEP adds configurable processing across an event stream.

Complex Event Processing in Zabbix

Rules, event context and time windows





Create complex event processing

Create event correlation



↓

Data collection ^

Template groups

Host groups

Templates

Hosts

Maintenance

Event processing

Discovery

New complex event processing ? x

* Name

My first rule

Conditions

Label	Name	Actions
A	Event name equals <i>Test event</i>	Edit Remove

Add

Time window ?

None

Simple (multiple)

Simple (once)

Cause and symptoms grouping

Pattern match

* Operations ?

Details	Conditions	Actions
1: Execute when Event occurred : Set name <i>My first correlated event</i>	Problem is opened	Edit Remove

Add

Stop after this rule

☒

* Sort order

0

Description

It works!

Enabled

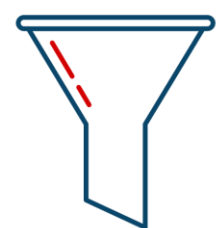
☒

Add

Cancel

Anatomy of a CEP rule

A rule defines which events matter and what processing they receive.



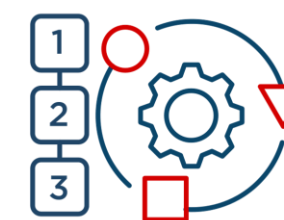
Select events

Name, severity, host context
and event tags



Choose a window mode

Individual events or
a grouped time window



Apply operations

One or more operations at
selected execution points

Rule order and "Stop after this rule" control which subsequent rules can apply.

Five time window modes

Choose how CEP handles matching events.



None

Run operations as each event arrives.
No events are kept in a time window.



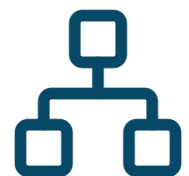
Simple (multiple)

Keep events for a duration; each event expires at its own event time plus duration.



Simple (once)

Use the same sliding window behavior.
Only the first matching rule of this type runs.



Cause and symptoms grouping

Make the first event the cause and later events its symptoms within a fixed period.

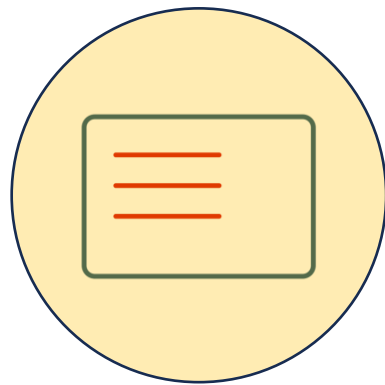


Pattern match

Run JavaScript on the current window to detect an event pattern.

Time window: None

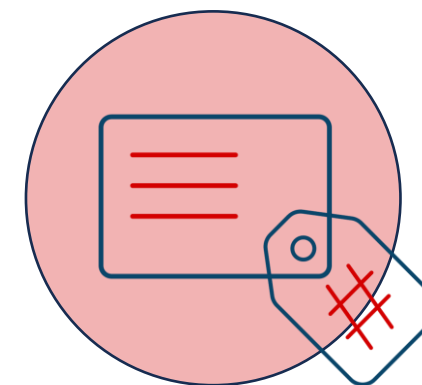
Process each matching event as it occurs.



Incoming event

service = checkout

Event occurred



Enriched event ↑

service = checkout

team = payments

Example operations: add the tag team = payments and increase severity

No duration, capacity or accumulated event context.

Time windows

Time windows keep matching events for a set time and runs operations as they enter or leave.

Time window ?

None	Simple (multiple)	Simple (once)	Cause and symptoms grouping	Pattern match
------	-------------------	---------------	-----------------------------	---------------

* Duration

* Capacity ?

Unlimited	Limited	10
-----------	---------	---------------------------------

Group by Host group, Host or Tag.

Group by

☒ Host group

☒ Host

☐ Tag ?

Simple time window

Keeps matching events for a set time and runs operations as they enter or leave.



Duration and expiry

Each event expires once the configured duration time has passed from its original timestamp. Events leave the window when they expire.



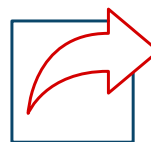
Capacity

Limits the number of events in each window.
When full, new events trigger "Event evicted" without joining.



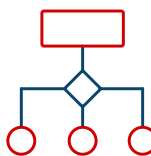
Event added

If there is space, the event joins the window.
"Event added" operations run.



Event evicted

"Event evicted" operations run when an event expires
or when a full window rejects it.



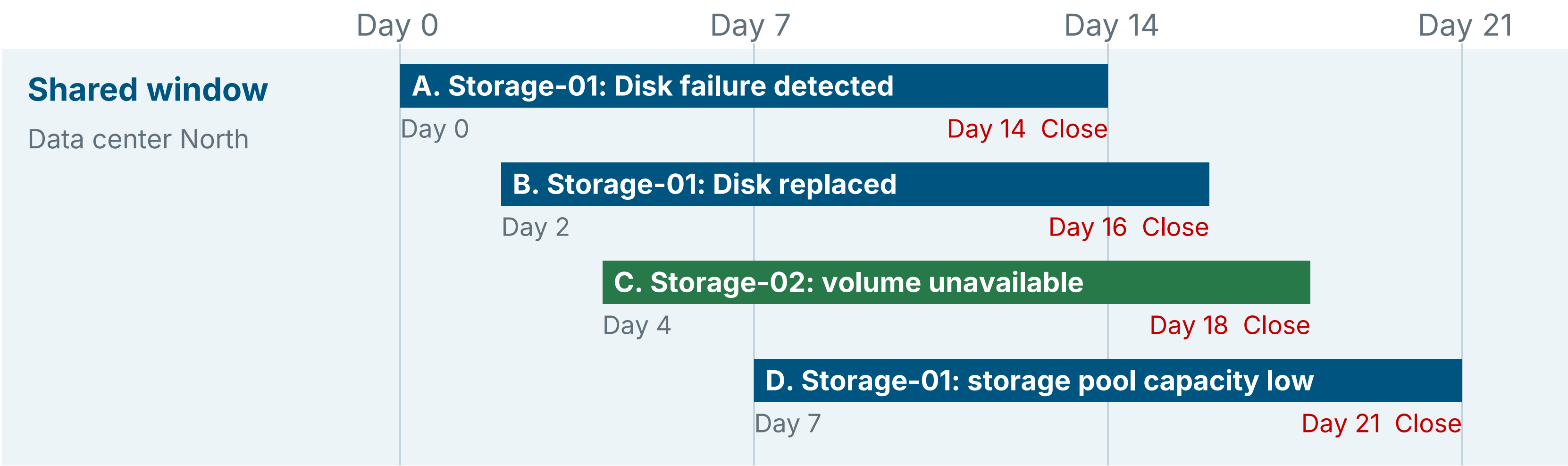
Grouping

No grouping: one shared window per rule.
With grouping: separate windows, each with its own capacity.

Simple window, **no grouping, no capacity limit**

Filter: dc = north

Duration: 14d Group by: none Capacity: Unlimited



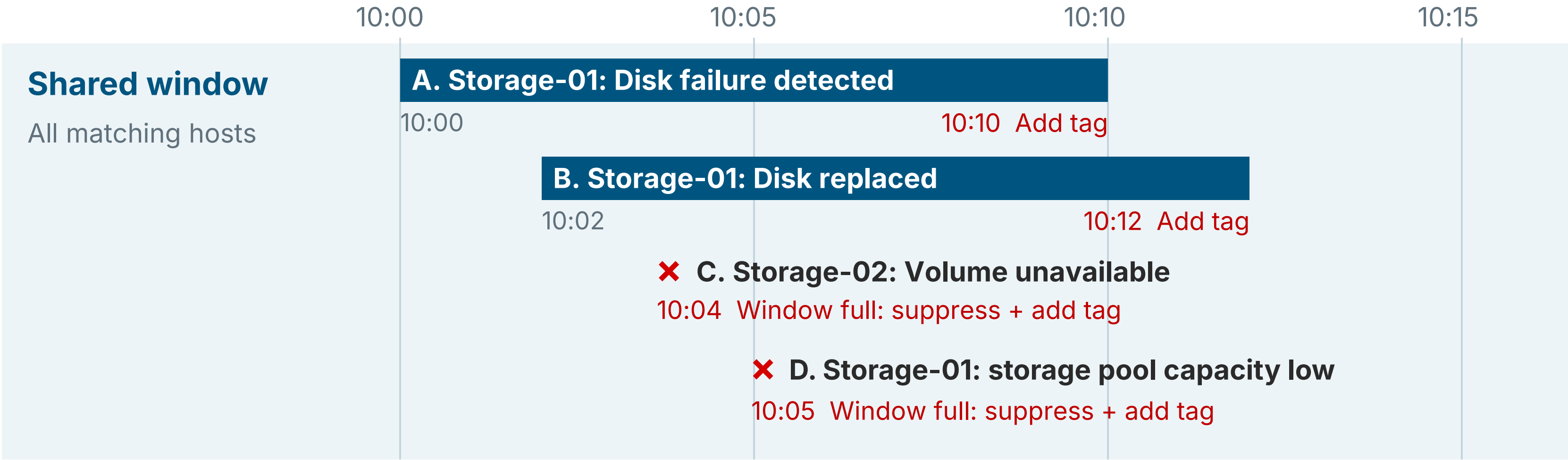
Event evicted: Close

Close still-open problems in data center North when they reach 14 days of age.

Simple window, no grouping, **capacity of two**

Filter: component = storage

Duration: 10m Group by: none Capacity: 2



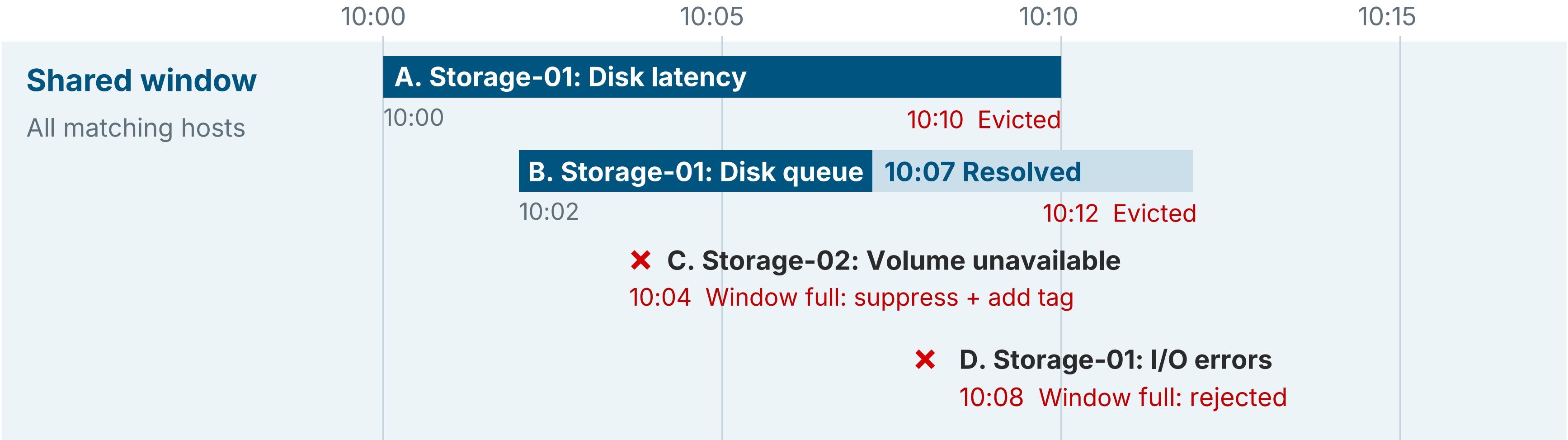
Event evicted: suppress and add tag cep = evicted

C and D run the operations on rejection. A and B run them later, on expiry.

Simple window, no grouping, **resolved events**

Filter: component = storage

Duration: 10m Group by: none Capacity: 2



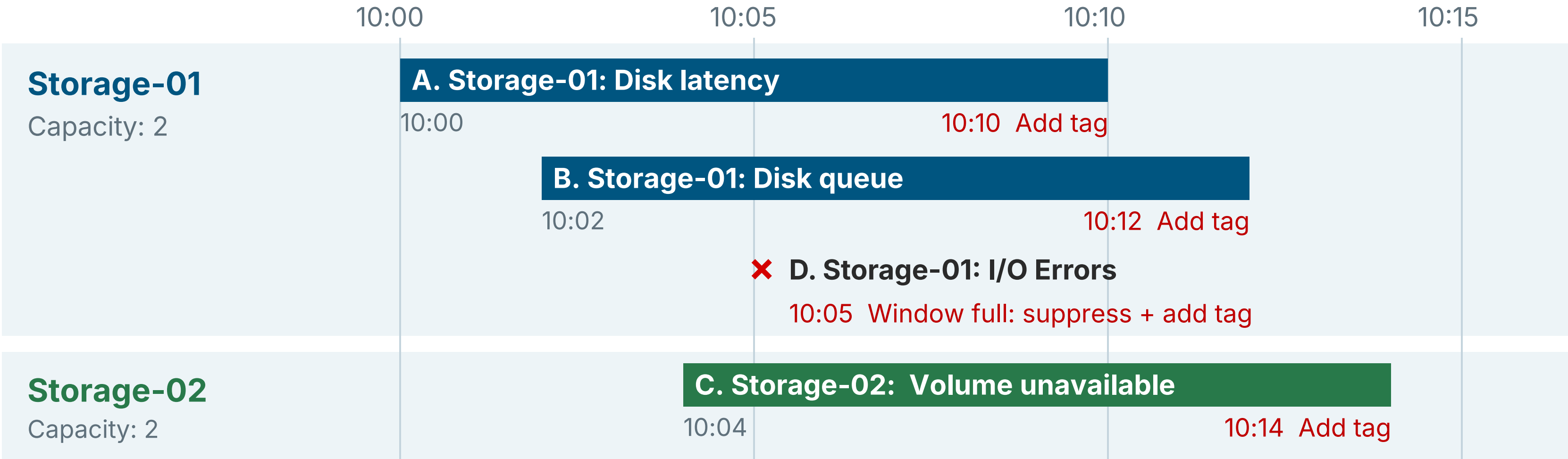
Event evicted: suppress and add tag cep = evicted

B remains in the window after recovery. C and D are rejected because both slots are still occupied.

Simple window, **host grouping**, capacity of two

Filter: component = storage

Duration: 10m Group by: host Capacity: 2 per window



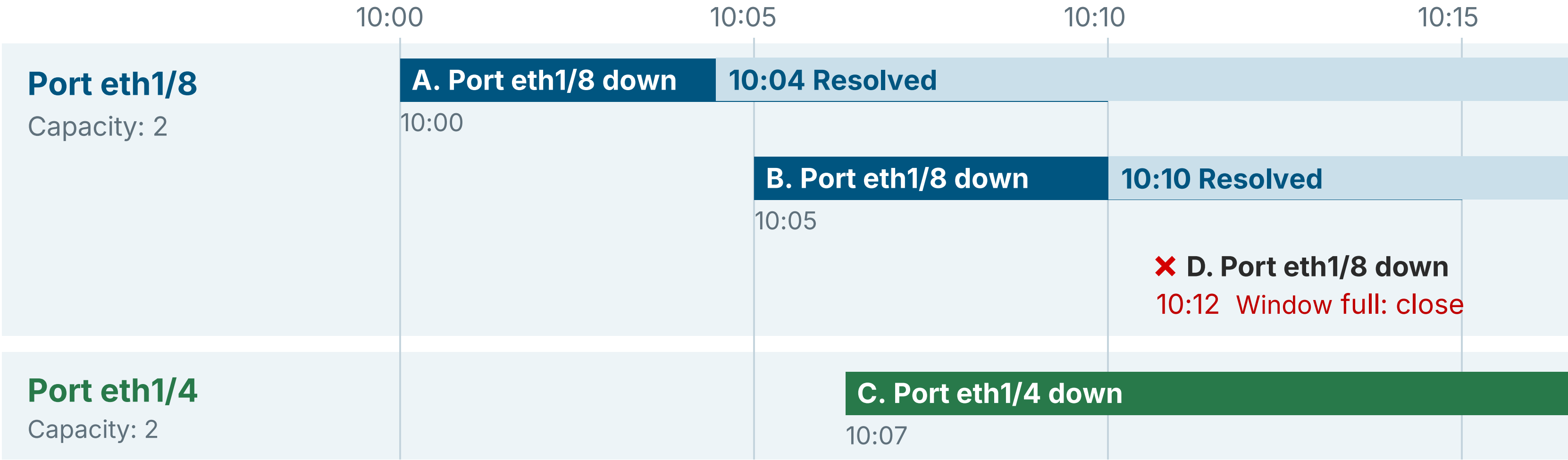
Event evicted: suppress and add tag cep = evicted

C joins storage-02, which has its own capacity. D cannot join storage-01 because it is full

Automatically closing flapping port alerts

Filter: component = ethernet port

Duration: 1d Group by: tag: port name Capacity: 2 per window



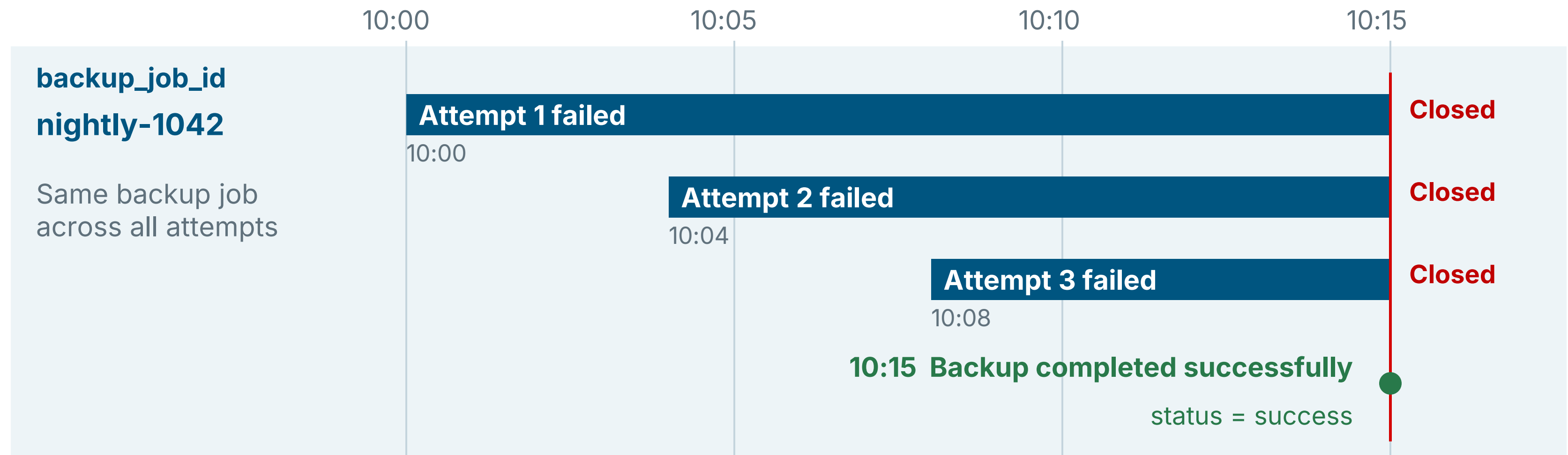
Event evicted: close

D cannot join port eth1/8 and is closed. C joins port eth1/4, which has its own capacity.

Backup retries: one success closes three failures

Filter: process = backup

Duration: 1h Group by: tag: backup_job_id Capacity: Unlimited



Event added, status = success: Close (window) Window closed: Close

The success event closes all three failures and itself. All four are separate log-generated PROBLEM events.

Simple (multiple) vs. Simple (once)

Both can match tags. The difference is how overlapping rules process an event.

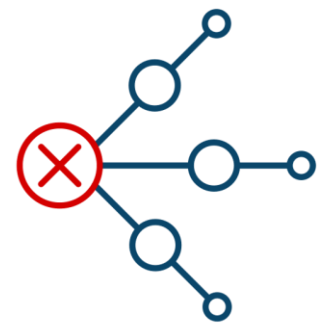
Capability	Simple (multiple)	Simple (once)
Group events by tag values	✓	✓
Use tag conditions in operations	✓	✓
Sliding expiry and capacity limits	✓	✓
Close a window through an operation	✓	✓
Process an event in multiple matching rules of this mode	✓	✗

✓ Supported ✗ Not supported

* "Stop after this rule" can further limit processing in either mode.

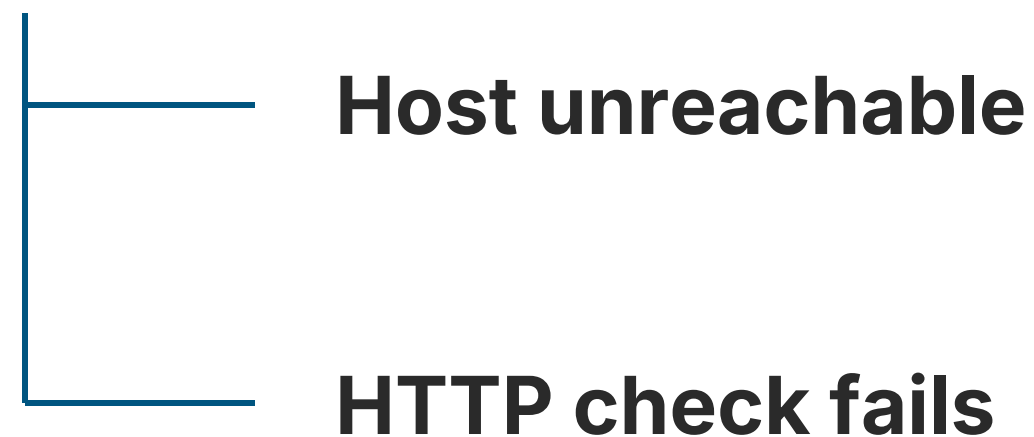
Cause-and-symptom grouping

The first event in the group becomes the cause.



Uplink down

First event in this group



Symptom

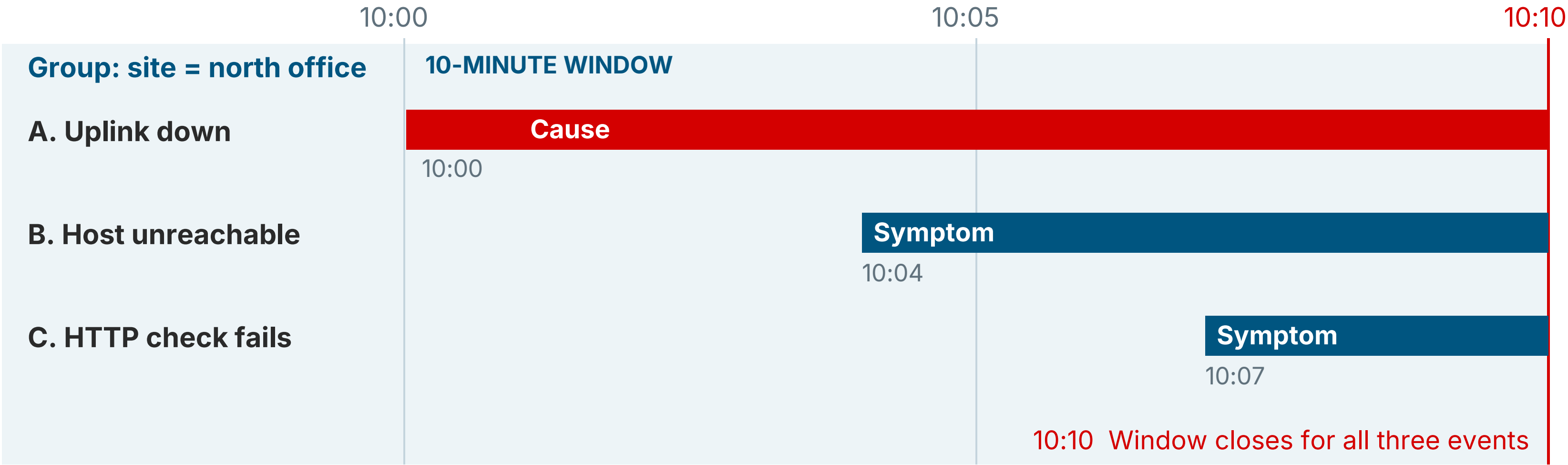
Symptom

Fixed time windows. Optional count tag on the cause. Arrival order is an assumption about causality.

Cause-and-symptom grouping

Filter: site = north office

Duration: 10m Group by: host group Capacity: Unlimited



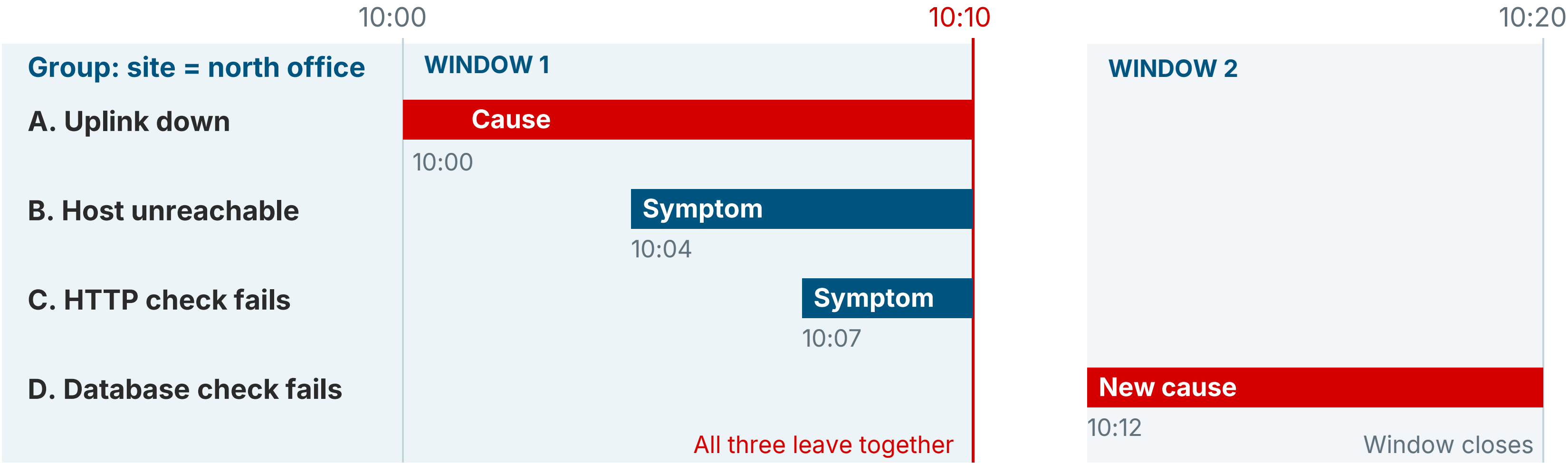
The first event becomes the cause. Later events become its symptoms.

Window closure ends membership. It does not automatically recover problems or remove their ranking.

Cause-and-symptom grouping: **Next cause**

Filter: site = north office

Duration: 10m Group by: host group Capacity: Unlimited



The first event in each fixed window becomes the cause.

Window closure ends membership. It does not automatically recover problems or remove their ranking.

Pattern matching: how JavaScript works

The script evaluates window events. Configured operations act on a match.



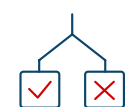
Input

`cep_get_events()` returns the events currently in this window.
Read their tags, names, severity, timestamps and status flags.



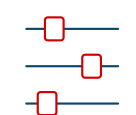
Evaluation

CEP runs the script periodically, after removing expired events.
JavaScript checks your conditions across the available events.



Output

Return `"true"` when the pattern matches, or `"false"` when it does not.
The script reads event data; it does not change the events directly.



Operations

A `"true"` result runs operations set to `"Pattern matched"`.
Each operation applies its own conditions and configured action.

A match can repeat at the next evaluation. Operation conditions must account for this.

Pattern matching use case

Three failed logins trigger an alert while the window stays open.

Window	Failed logins by source_ip. Duration: 5m. Capacity: Unlimited.
Pattern	Count ≥ 3 original events, including suppressed ones. Exclude copies.
When matched	Clone first if original and unsuppressed, then suppress all originals.
Cloned event	Set name to "Brute force attack" and severity to High.
Window stays open	Events keep their own expiry times. The copied alert stays unsuppressed.

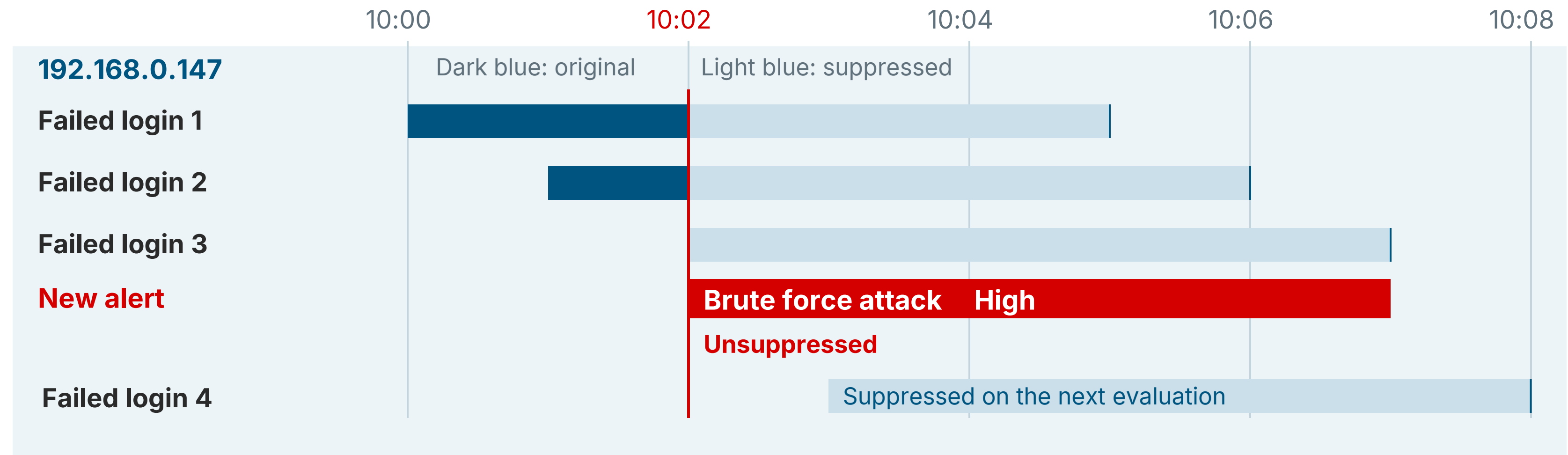
Clone before suppression. A suppressed first event cannot be cloned again.

Further originals are suppressed on evaluation while the count remains ≥ 3.

Clone the first event, suppress the originals

Filter: `event_type = login_failed`

Duration: 5m Group by: `source_ip` Capacity: Unlimited



At 10:02: clone first and suppress all three originals. At 10:03: suppress the next login.

Window stays open. Bars end at expiry, which does not close problems or end suppression.



* Name

Conditions	Label	Name	Actions
	A	Host contains <i>Linux</i>	Edit Remove
	Add		

Time window ?

None

Simple (multiple)

Simple (once)

Cause and symptoms grouping

Pattern match

* Duration

* Capacity ?

Unlimited

Limited

Group by

☐ Host group

☒ Host

☐ Tag ?

* Script

var events = cep_get_events();...

* Operations ?	Details	Conditions	Actions
	1: Execute when Event occurred : Set name <i>Brute force attack</i>	Event is cloned	Edit Remove
	2: Execute when Event occurred : Set severity <i>High</i>	Event is cloned	Edit Remove
	3: Execute when Event occurred : Unsuppress	Event is cloned	Edit Remove
	4: Execute when Pattern matched : Clone first	Event is not cloned Problem is not suppressed	Edit Remove
	5: Execute when Pattern matched : Suppress <i>Indefinitely</i>	Event is not cloned Problem is not suppressed	Edit Remove
	Add		

Stop after this rule ☐

* Sort order

Description

Detects brute force after 3 unsuccessful logins, creates a new event and suppress next events

Enabled ☒

What **CEP** operations can do



Event name

Set name



Event tags

Add, set, rename, remove or adjust values



Problem closure

Close an open problem



Severity

Set, increase or decrease severity



Suppression

Suppress or unsuppress a problem



Event and window processing

Discard, clone first / last, close window

Tag operations

Choose how a rule changes an event's tags.



Add

Add a new name/value pair.

Example: site=north



Set value

Update an existing tag only.

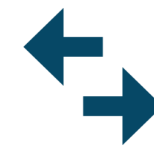
Missing tags stay missing.



Rename

Change the name. Keep the value.

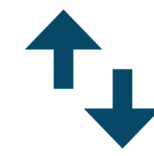
team=ops becomes owner=ops



Set

Create a tag or update its value.

Example: team=platform



Increase / Decrease value

Adjust an existing numeric value by 1.

attempts=2 becomes 3 or 1



Remove

Delete an existing tag.

team=ops disappears

Choosing a **CEP** mode

Mode	Use it when you need	Example
No window	An operation on each event	Add a tag and increase severity
Simple (multiple)	Operations on window entry or eviction	Apply separate retention or processing policies
Simple (once)	Related events to complete a sequence	Assign an event to the first matching handling rule
Cause and symptoms	First-event cause-and-symptom ranking	Group a burst of events at one site
Pattern matching	Custom logic across events	Detect a combination of failures

Training courses





Core Course | Level 3

Zabbix Certified Professional

Deploy a Zabbix environment from scratch while learning how to automate your monitoring workflows and scale your Zabbix instance with high availability and load balancing in mind.

Duration: **3 days**

Course prerequisite: **Zabbix 8.0 Certified Specialist exam or attendance certificate**



Available as **live training**



Extra Course | Level: Intermediate

Problem and Anomaly Detection with Zabbix

This course aims to teach you how to prevent alert fatigue, populate your alerts with meaningful insights, and help you focus on what truly matters by correlating detected problems with their root causes.

Duration: **1 day**

Course prerequisite: **None**



Available as **live training** and on **Zabbix Academy**

Want to dive deeper into Complex Event Processing?

Workshop



CEP workshop

Zabbix Summit 2026

13:00 – 14:00

Hands-on complex event processing

Cloud virtual machines provided

All five time-window modes

-  None
-  Simple (multiple)
-  Simple (once)
-  Cause and symptoms grouping
-  Pattern match

Everyone is welcome

Thank You!

