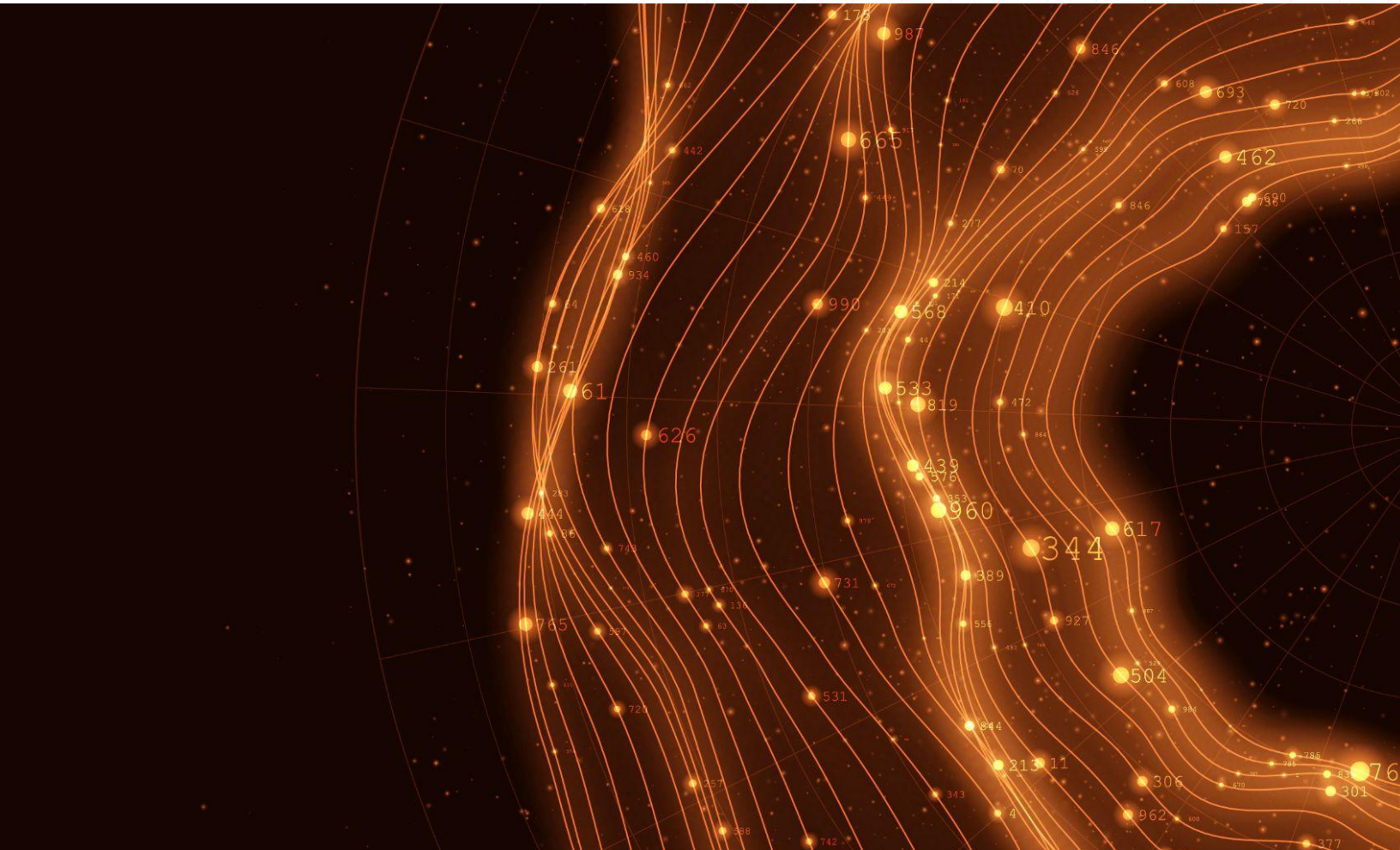
ctrlstack

Whitepaper

Introducing CtrlStack: A Better Way to Observability.

The DevOps observability platform that connects cause to effect for real-time troubleshooting.



Whether you're just starting with observability or further along in your process, the main goals across organizations are better service reliability, improved customer satisfaction, and faster troubleshooting - to keep pace with development speed. When delivering customer experiences from the cloud, reducing Mean Time to Recovery (MTTR) is a key driver for better observability. However, traditional monitoring solutions fall short. A 2022 study by Splunk found that the median downtime (MTTR) lasts more than 5 hours. This isn't good enough. It is a critical and expensive operational problem we're facing today.

The problems we solve.

The traditional method of looking at metrics, combing through logs, and attempting to pivot off of the data, can take several hours or days to determine the causes and explain how the event occurred. This is ineffective for troubleshooting cloud applications, and is a result of these lingering problems:

- **Monitoring tool sprawl** - Hunting down relevant data in multiple tools requires expertise in different UIs and query languages, and interpreting data returned in various formats.
- **Disconnected data** - Operational data used for observability (metrics, logs, traces, events) is helpful and necessary, but current tools don't allow us to view them as connected sources of information.
- **Fragmented knowledge** - Every DevOps engineer keeps a knowledge graph of all the infrastructure and interconnected service inside their heads; this graph is fragmented and takes time to learn and document, and mental power to retain.

While legacy monitoring tools are quickly evolving into "observability" tools by bolting on access to full datasets including logs, metrics, distributed traces, and events, the problems remain unresolved. The challenge is that operational data isn't just disconnected; it's dynamic and growing as the applications and the supporting infrastructure change. Another challenge is that the vast majority of operational incidents (75%) are caused by intentional changes - from deploying code, to changing configuration files, to provisioning services. Most downtimes are spent looking for the change culprit, and most organizations can't find that change. Why not fix the problem at its core and provide a solution that links performance issues to change – one that actually connects cause to effect in real time?

Why CtrlStack?

Backed by top-tier investors and an engineering team with extensive experience in the monitoring and observability space, CtrlStack's mission is to unify DevOps tools and knowledge to escalate software's impact. Starting with a clean slate, the team took a first-principles approach to troubleshooting for the fastest way to "why."

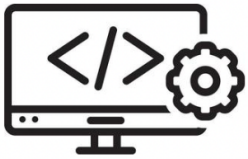
CtrlStack is the first platform that brings a unified experience for connecting cause to effect as changes happen. DevOps engineers can proactively track code deployment and configuration changes in a unified timeline, and connect those changes to operational outcomes - metrics, logs, and events. When an incident occurs, teams can trace backward in the timeline to investigate the cause(s) quickly. With a visual graph of the application and the infrastructure it sits on that can model cause and effect, teams no longer need to process the hidden connections in their heads. Sharing and documenting knowledge consistently across teams becomes effortless. Through these capabilities, CtrlStack lets teams add real-time change impact and root cause diagnosis into their DevOps pipeline.

Add Change Impact and Root Cause Diagnosis into your DevOps Pipeline

DevOps & SRE

Resolve issues faster with snapshots for every change

Developers



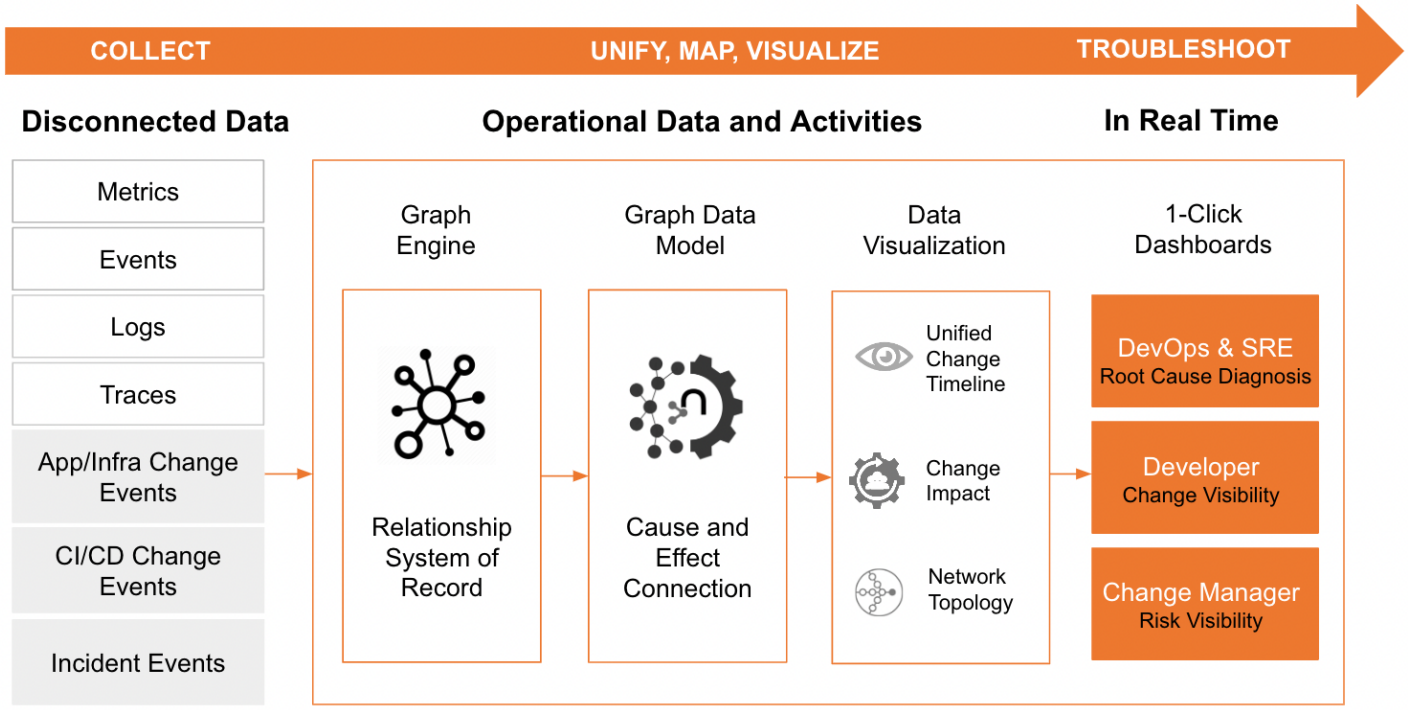
Proactively troubleshoot and debug code changes

Change & Release Manager

Reduce risks and deliver reliable services at scale

CtrlStack Graph-Based Troubleshooting.

The following diagram shows the key components of the CtrlStack architecture, including data collection, the graph engine that connects disconnected data, data modeling that maps changes to operational outcomes, and a unified DevOps experience for real-time troubleshooting.



Real-Time
Graph Construction.

In order to connect cause to effect and speed MTTR, DevOps teams need the ability to identify the cause of issues and see the blast radius of a system change. To accomplish this, incident response technologies need to construct a real-time graph of the application, infrastructure, and 3rd party services and update it every few seconds to incorporate new data and structural changes. While graph structures can be represented in many different storage layers, including SQL-based relational databases, a dedicated graph database offers many advantages – including performance and extensibility.

Most relational databases are “row stores”, optimized for storing and accessing **records** where many fields within each record will be accessed together. Row-oriented databases are sometimes contrasted with “column stores”, which are optimized for storing/querying data by **fields** (across multiple records). Graph databases are a third paradigm which are optimized for storing/querying data across multiple **relationships** between records.

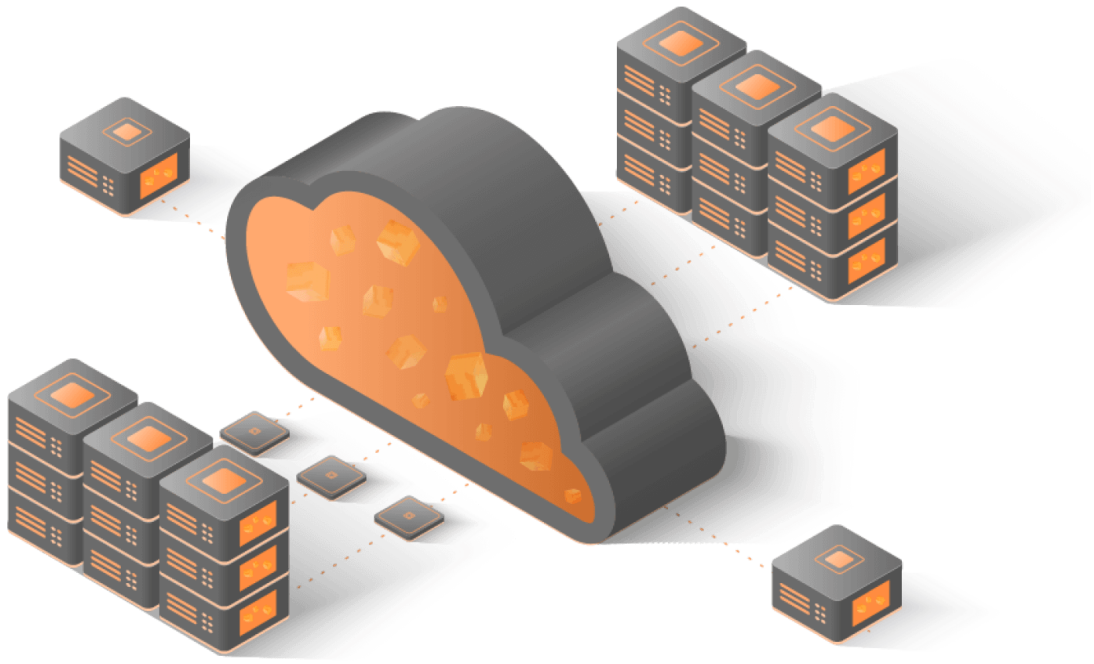
With a typical RDBMS, representing a relationship between two existing tables typically requires a third associative entity (“join”) table, with references to the primary keys of each target table, and with its own indexes. As that intermediate table grows, the performance suffers proportionally. Graph databases deliver higher performance for typical graph operations by storing relationships directly with each node, which can point directly to other nodes (this is known as “index-free adjacency”).

Graph databases also provide extensibility of new types of nodes, relationships, and graph architectures without any DDL schema updates. This extensibility is a must when dealing with modeling software and service systems, which are constantly evolving at both the macro scale (new types of technology) and at the micro scale (the relationships and connections of a particular system changing from one snapshot to the next, just a few seconds later).

The CtrlStack Native Graph Approach.

Metrics, events, logs, and trace (MELT) data that are collected by traditional observability tools are not sufficient for representing a real-time graph of a DevOps environment. The relationships between these raw data points - the data between the data - cannot be represented accurately with MELT and traditional tools. Even traces (which resemble graph relationships) are constrained to network requests, not other non-request network traffic, much less non-network dependencies – such as the critical relationships between a process running in a container running on a node in a Kubernetes cluster.

A common legacy technique to represent relationships is to use tags, such as “cluster=west01“ or “service=web-api”. However, tags are a poor approximation to a graph database; they have to store (and duplicate) the full list of connections with every copy of every data point, and cannot be easily or quickly traversed beyond a single hop. Tags make every data point into an effective node with slow-lookup relationships to other nodes.



Real-Time Relationship Mapping.

CtrlStack uses a native graph database to represent “the data between the data” – the relationships between MELT data. This enables every cloud application to be represented as a graph of components and their relationships and interactions. A native graph database approach achieves high performance and allows each data type (log, metric, etc) to be associated with a given node using a single relationship, leveraging all of the rich relationship data already existing with that node.

CtrlStack uses a variety of agents to construct the real-time system graph:

- **Cloud Agent** - Queries the raw inventory of entities/services running in a given account, as well as the cloud-native logs and metrics observability data available (AWS only).
- **Cloud Agent** - Host Agent: Runs as a process within a Linux operating system, and is optimized for EC2 instances within AWS. It queries the process table and helps to construct a network graph.
- **K8s Agent** - Runs as a Daemonset within a Kubernetes or EKS cluster, and collects detailed K8s-specific information not available from the Cloud APIs.
- **Service Agent** - Handles 3rd-party service providers, such as GitLab, Split.io, LaunchDarkly, Snowflake, and Databricks - these services form an extended portion of your system, and are directly connected to the internal system topology.

The different agents pull different types of data (both local **subgraphs** and also **non-graph data**, such as metrics and properties) back to the CtrlStack modeling engine, which assembles and identifies matching nodes across different subgraphs to tie them together into the canonical **supergraph**. This supergraph is assembled from scratch every few seconds, so that changes can be recognized quickly for operational control, and to ensure a high-resolution record of historical graphs for troubleshooting, root-cause analysis, and post-mortem reporting. CtrlStack stores these historical snapshots in AWS Elastic File System (EFS) and allows teams to retrieve them programmatically and through the UX.

Actionability in Troubleshooting Flows.

CtrlStack provides two out-of-the-box troubleshooting flows to get new users started quickly: change impact and root-cause analysis. These read-only flows were designed to safely introduce users to CtrlStack so the ability to take actions from these flows are disabled.

While these read-only flows will deliver immediate value to customers, they are just the beginning for accelerating incident response. To enable users to act on the data available in the graph, CtrlStack attaches the **actions** for each entity to the corresponding node. For example, every EC2 instance node within CtrlStack has both relevant metrics attached to it as well as actions such as TerminateInstance. To take actions on any entity, users can easily navigate to the dedicated topology pane from the troubleshooting flows/panes.

Fast Data Query for Faster Troubleshooting.

CtrlStack's powerful StackQL Language allows teams to query the graph data structure efficiently. By combining graph query operations with typical log/metric query patterns, users can query current snapshots and historical snapshots to compare current with previous system states for faster investigation. Internally, StackQL, a python library, is transpiled to Clojure, which assembles the data sources across the graph database, metric store, logs, and events management into a single JSON payload for display and/or processing.

System of Record for Change Management.

DevOps teams today lack a centralized management hub for system changes that happen in production. This system of record is needed to show teams when, where and why changes are made, and how those changes impacted operations. In fact, a recent survey found that an overwhelming 76% of all outages can be traced back to changes in a system's environment. Without the ability to track changes in production, teams cannot correlate changes with incidents and outages.

CtrlStack helps DevOps teams manage a wide variety of operational activities and sources of changes to reduce risks, track change impact, and find root causes of production issues fast. CtrlStack integrates change events from many different sources and then stores them in a common format for analysis. Teams can quickly explore and filter this data using a real-time timeline and a network topology - all in one place.

The following are some of the sources of change events currently managed by CtrlStack.

Events	Description
AWS CloudTrail	CloudTrail records users activity and API usage in AWS services. When a user or external system makes a call to an AWS API in your AWS account, a CloudTrail event is emitted.
AWS EventBridge	EventBring is a serverless event bridge that receives events from AWS Services, including CloudTrail, as well as your own applications and micro services. Using EventBridge, a user can configure their internal EventBridge service buses to route events to CtrlStack's EventBridge Service Bus (all within AWS). This allows CtrlStack to analyze the firehouse of events occurring within your AWS environment for change impact analysis.
Kubernetes Events	Like AWS, a moderately sized Kubernetes cluster can emit a firehose of events. Events are emitted

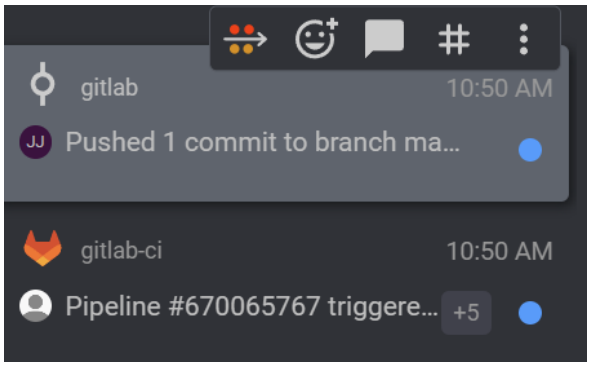
	in Kubernetes when state changes, which can be frequent in an actively developed environment. When changes occur in your Kubernetes cluster, the emitted events are tied into CtrlStack's knowledge graph to show the impact of kubernetes events on the Kubernetes cluster as well as externally connected system (ie. AWS).
Config Files	CtrlStack monitors file systems and collects and stores events that indicate key files change on disk. Changes to configuration files can be a cause of a failure (ie, service no longer starting). Capturing file change events and presenting them immediately to a DevOps engineer investigating a problem can significantly reduce the time required to identify the root cause of a problem.
Terraform Files	Terraform is widely used to make infrastructure changes. Changes to terraform files trigger calls to AWS APIs, which then emit CloudTrail events. CtrlStack captures changes to terraform state, associates that change with a logical entity and then ties those changes to impacted events. This allows for engineers to identify a change in infrastructure and trace it back through the AWS API call (CloudTrail) to the terraform change that triggered it.
PagerDuty Events	CtrlStack integrates with PagerDuty to collect events that are usually indicative of a problem in production. If the events have information pointing to the infrastructure in place, CtrlStack can tie these events into its root cause analysis system. This provides complete symptom traceability, allowing users to navigate from a pager duty event to the impacted infrastructure, to the AWS API call, to the changed terraform file, and back to the git commit.

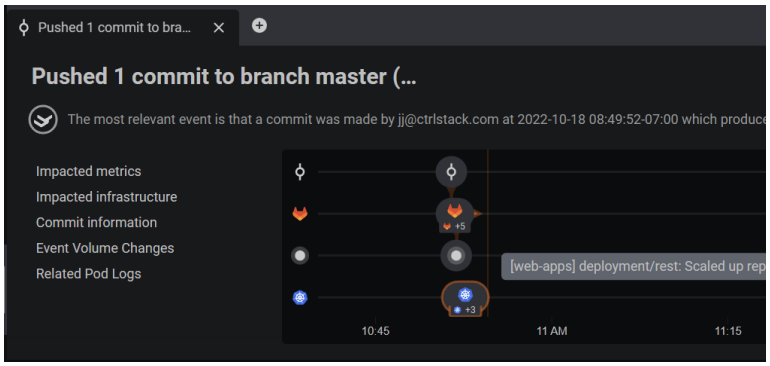
SSM/Terminal Commands	Many systems running in non-container environments are managed via SSM and terminal access. CtrlStack captures the commands that are entered by a user within SSM and/or a terminal and collects these commands as events. Tying the commands to events allows CtrlStack to connect SSM commands with impacted infrastructure as well as AWS/K8s API calls (which trigger events).
CI/CD Code Deploys	CtrlStack monitors your Git repositories and collects events when new commits are made. Many times, new commits will trigger CI/CD pipelines which can lead to deployments that negatively impact your infrastructure. CtrlStack captures these CI/CD pipeline events which can be traced back to Git commits, and displays the changes in the code. CI/CD pipeline events can also be traced forward to view impacted infrastructure and other related events that were triggered as a side effect of code deployment.

Change Impact Dashboard.

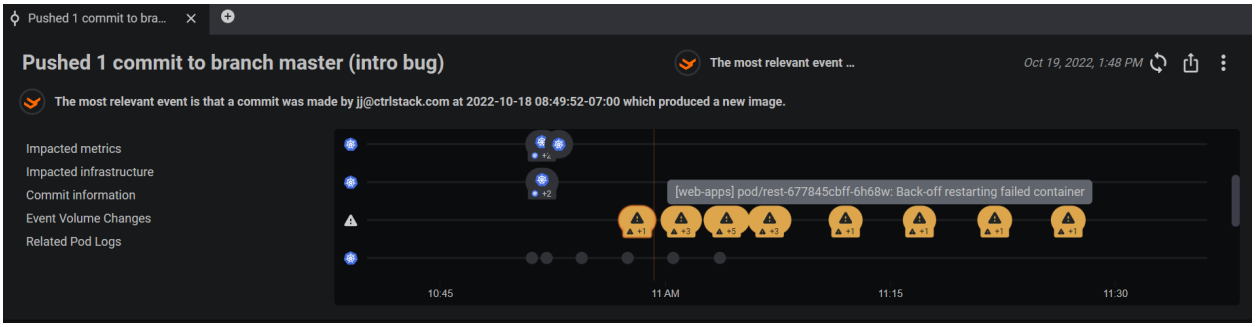
Intentional changes, such as committing and deploying new code, are oftentimes accompanied by change anxiety. The Change Impact dashboard provided by CtrlStack transforms anxiety into confidence by delivering a simple, yet powerful way for engineers to start with a potential cause (ie - deploying new code), identify associated events, and be presented with critical troubleshooting information should any negative effects occur.

The Change Impact dashboard can be launched from an associated commit found in the timeline.





The Event Timeline, which displays connected events to your code deploy, provides a clear path of activity that occurs from deployment to production readiness. Developers are able to quickly spot expected activity, such as a git commit triggering a pipeline build or scaling of replica sets in their Kubernetes deployment. More importantly, they will see critical, potentially unexpected activity such as their pods going into crashloopbackoff.



When an unexpected error occurs, engineers attempt to find and follow a trail of breadcrumbs in the hopes that it leads to the cause(s). The time it takes to investigate these clues, which often come in the form of metrics, logs, and reviewing

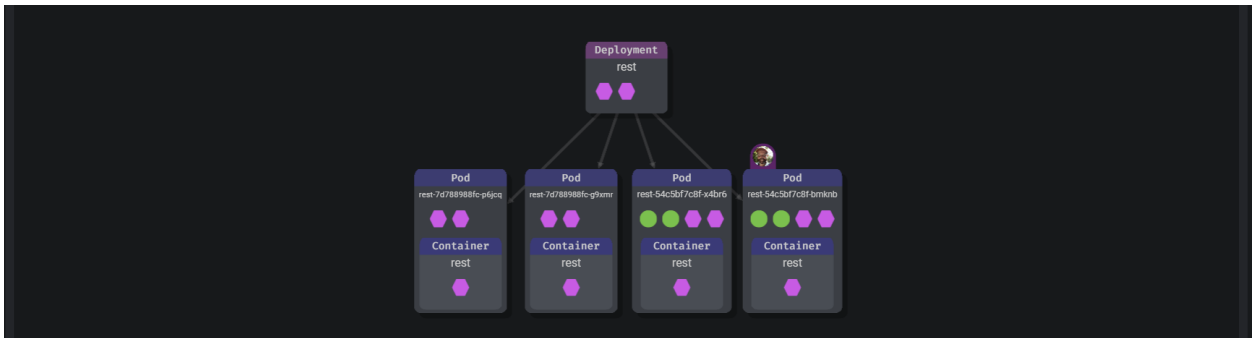
commit versions, only causes the change anxiety to grow. The Change Impact dashboard generates critical clues in a unified way so it's easy to identify the cause(s) and plan the proper steps for resolution. These clues begin with Impacted Metrics.

Observing metrics behavior post-deployment is a common way to validate whether a recent deployment has any potential side effects to investigate. However, the ability to quickly diagnose health via metrics behavior is slowed down due to several problems:

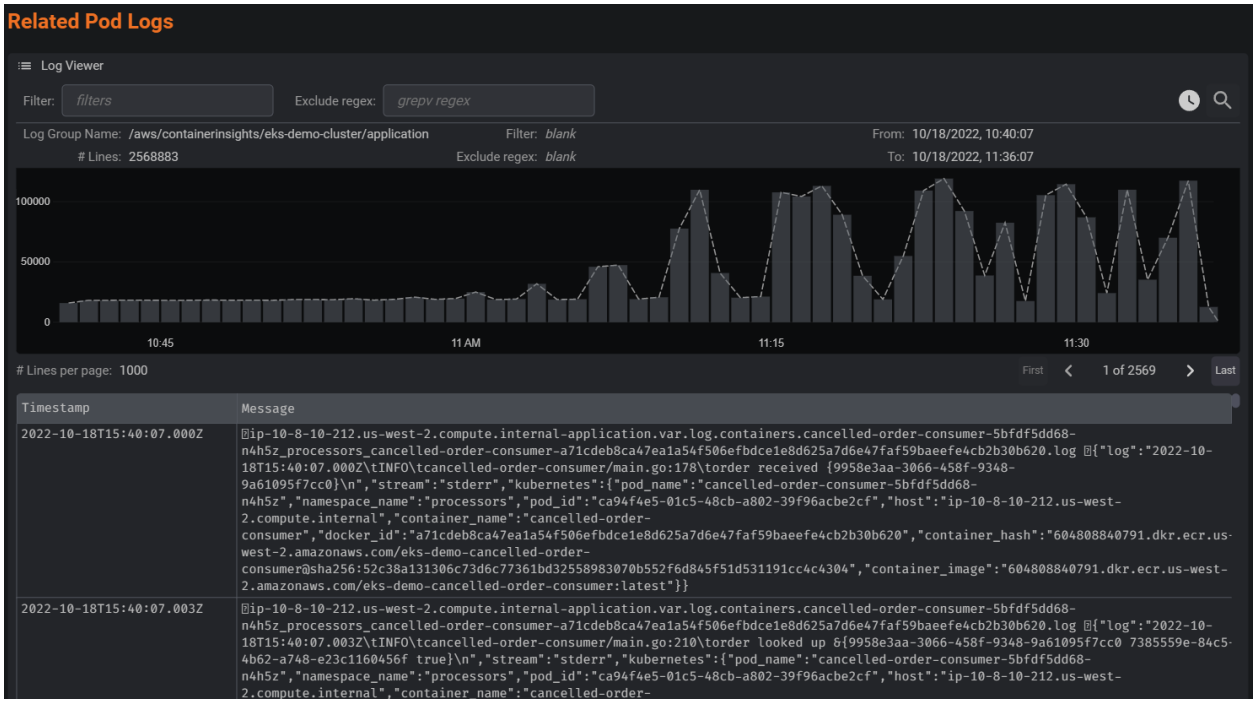
- Needing to know which metrics to observe
- Finding where they need to be observed from
- Human error potentially missing anomalous behavior.

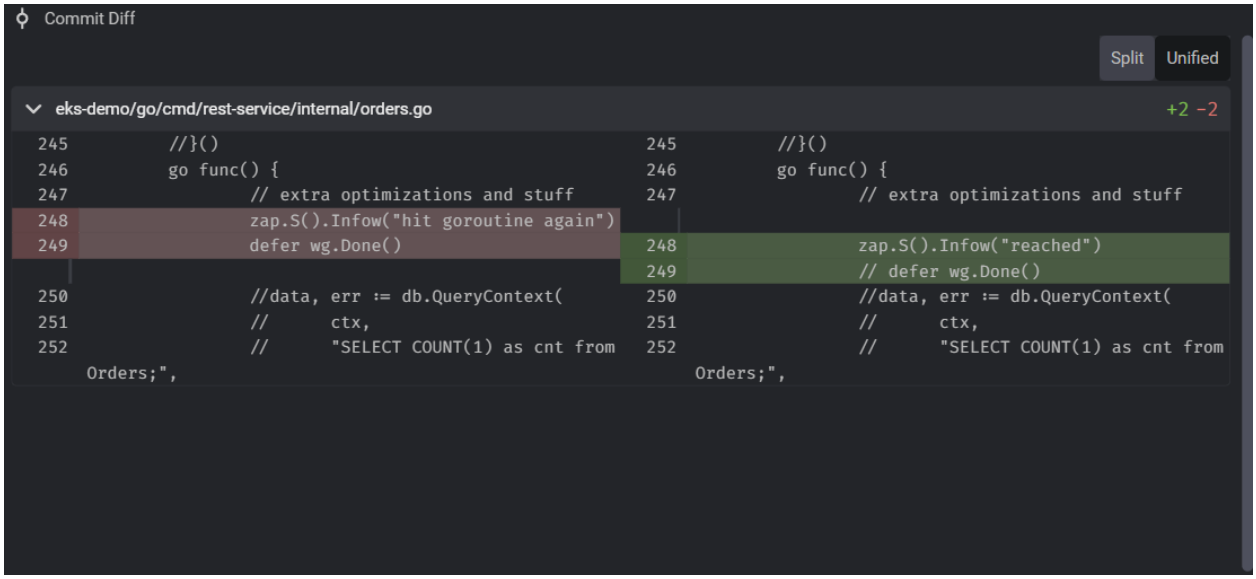
Through real-time relationship mapping, the Change Impact dashboard is able to immediately identify which metrics to observe, and does anomaly detection on those metrics automatically. Each metric with anomalous behavior is flagged and displayed as their own chart widget so users can observe the behavior change directly.

The Impacted Infrastructure helps to identify which components in a stack are associated with the selected commit.



Logs can also be a helpful tool when debugging a code commit. A particular log message appearing and how often it appears can potentially point to the root cause. With Related Pod Logs, the Change Impact dashboard delivers this helpful information directly next to the Commit Diff information, enabling users to spot an issue and review the code commit in a unified way.





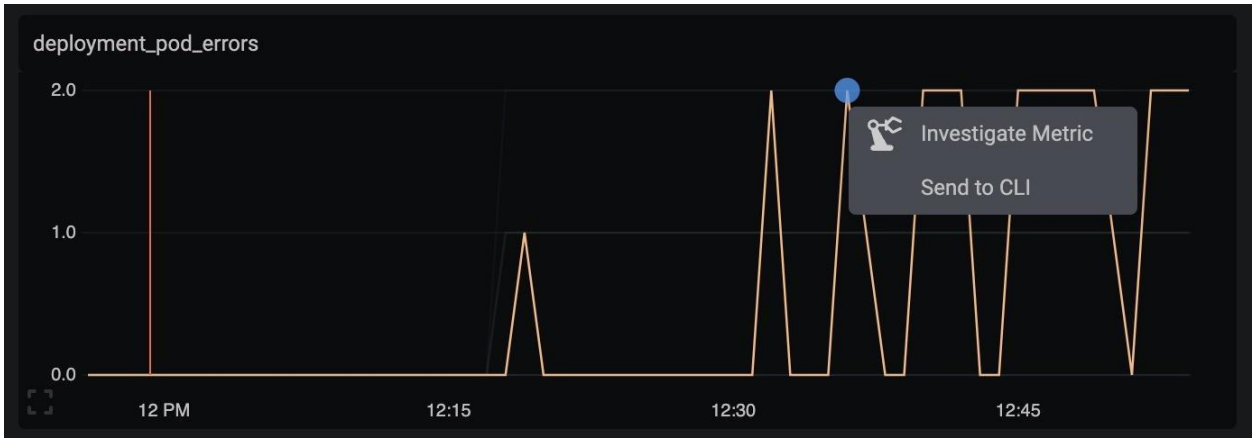
Root Cause Analysis Dashboard.

The Change Impact flow described above connects a known cause to unknown effects. CtrlStack's Real-Time Troubleshooting flow is a similar capability which goes in the other direction, from a known effect/symptom to the unknown, underlying cause(s). Because about two-thirds of unplanned downtime is spent trying to identify the root cause of the issue, this capability is especially leveraged during incidents. If Change Impact answers the "How" (does this change affect the system), Real-Time Troubleshooting answers the "Why" (did this behavior occur), tracing behavior back to change event(s) for faster remediation.

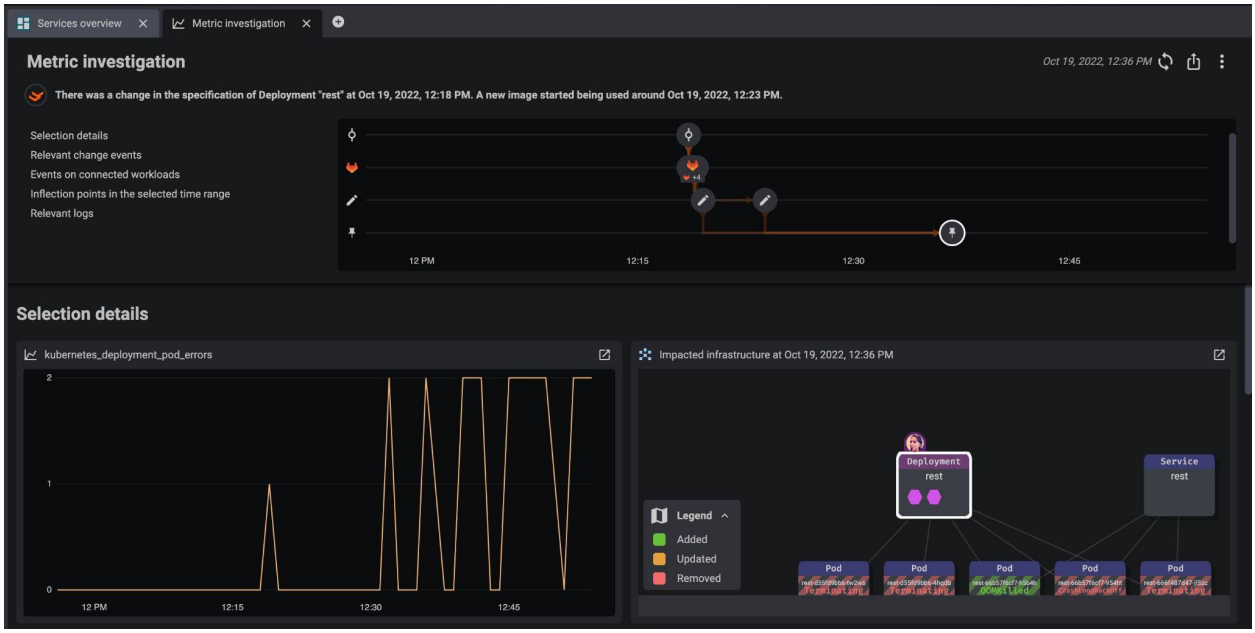
Like the Change Impact flow, Real-Time Troubleshooting uses the cause/effect graph tying together operational data to hone in on the most likely causes (based on which components directly affected other components), not simply the most coincidental causes (based on which events happened around the same time).

There are two ways to trigger the Real-Time Troubleshooting flow: from a metric chart, or from an event in the event timeline.

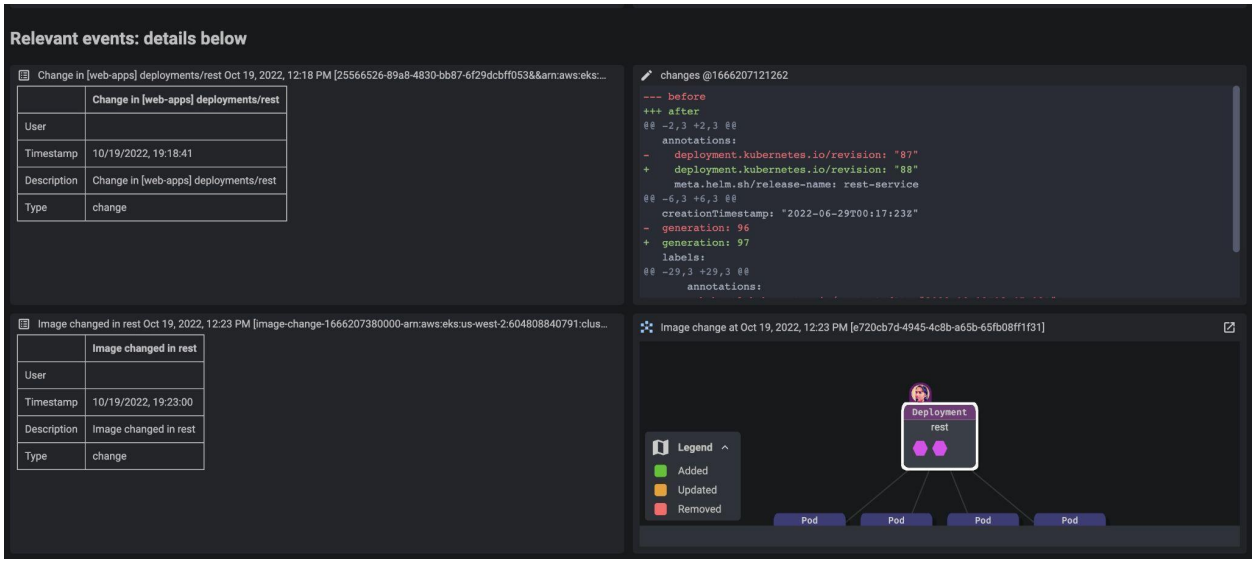
Metric charts often show unusual or unexpected behavior that require deeper investigation; for example, a spike or a trend in a typically stable metric, or a baseline change. In this case, the unusual behavior can be identified and selected with a right-click:

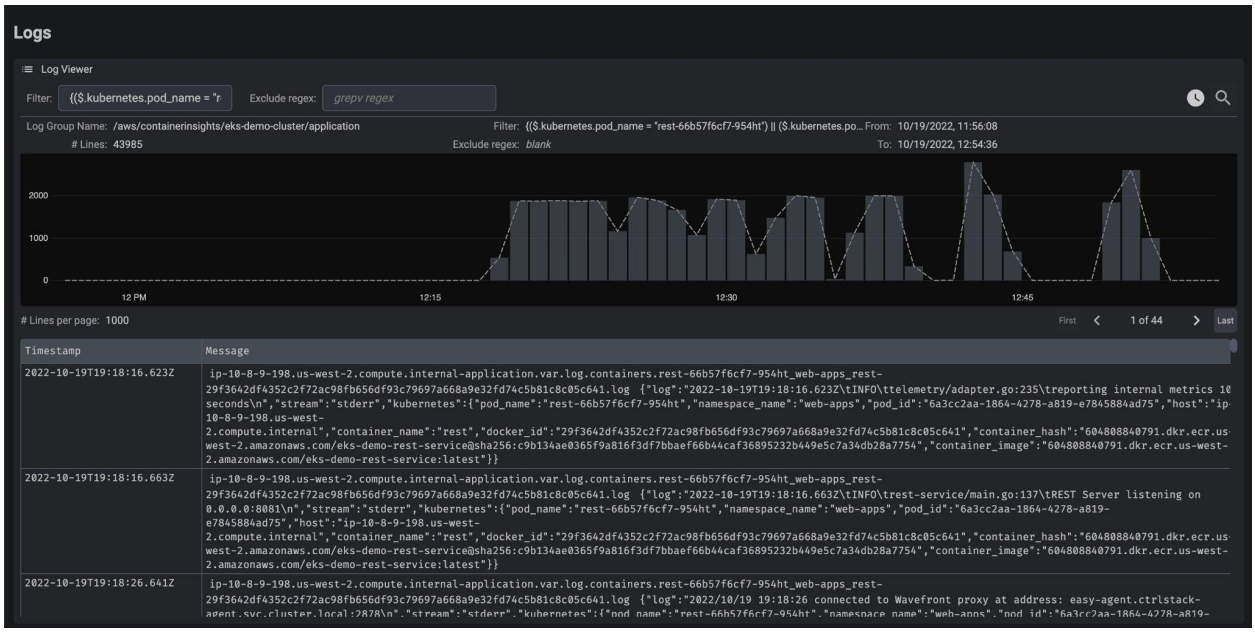


Clicking “Investigate Metric” will immediately trigger the real-time troubleshooting workflow in a new tab, allowing you to view the history of changes leading up to the behavior. For example, in the case below, a commit triggered a build pipeline, which then deployed images to the cluster, resulting in the metric change (deployment_pod_errors) that began the investigation.



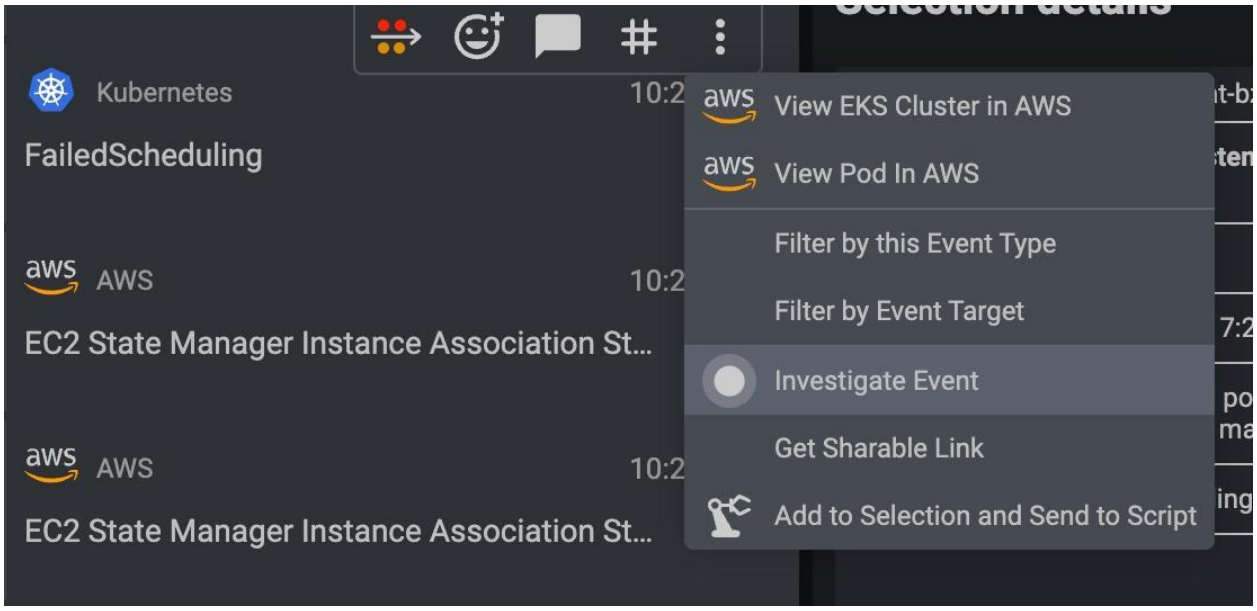
The Metric Investigation workflow also shows the impacted machines in a topological view (at bottom right), as well as the commit diff itself, detailed pipeline information, and relevant log streams.





All of this information can be retrieved in seconds during the incident itself to assist in real-time remediation, or after the fact to help flesh out a post-mortem analysis.

Event Investigations are initiated in a similar way by clicking the ‘Investigate Event’ option within the 3-dot hamburger menu at the top-right of each event.



This flow will also open a new tab in the primary window, showing the initiating event as well as previous events which have a causal relationship with that event.

About CtrlStack

CtrlStack provides incident orchestration to prevent downtime and identify changes before they impact customers. The platform unifies disconnected data, teams, and tools to connect cause to effect in DevOps. Harnessing the power of those connections, CtrlStack enables automated troubleshooting and serves as the system of record for change management. For more information, visit ctrlstack.com or join the conversation on LinkedIn, Twitter and YouTube.