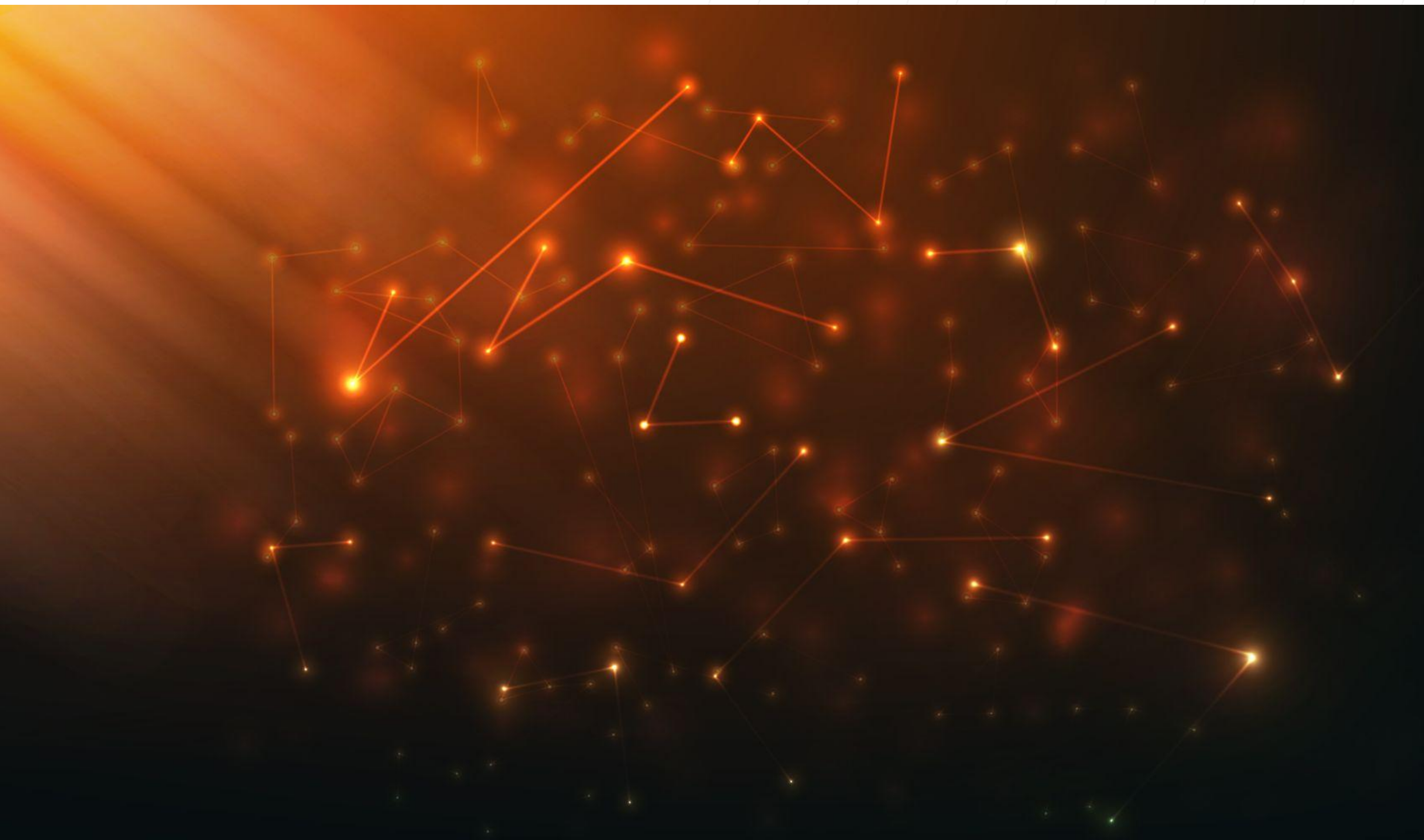
ctrlstack

Whitepaper

Rethinking DevOps as a Graph for Real-Time Troubleshooting.

Find connections to root causes and resolve incidents faster.



Executive summary

For enterprises adopting microservice architectures and continuous deployments, it is crucial to efficiently diagnose the root causes of incidents to maintain service reliability and uptime. The scale and complexity of distributed systems with microservices and continuous deployments can hinder observability. It’s becoming increasingly difficult to trace an error back to the specific change that caused it. How do we know if an operational issue is related to code, infrastructure, or operational activity?

Every DevOps engineer naturally keeps a knowledge graph of all the infrastructure and interconnected services - inside their heads. These manual connections require familiarity with the system and substantial experience in operation and management. But today’s complex and dynamic architectures make it impossible to keep up with all the changes around them. This cognitive load is compounded by siloed monitoring tools that increase context switching.

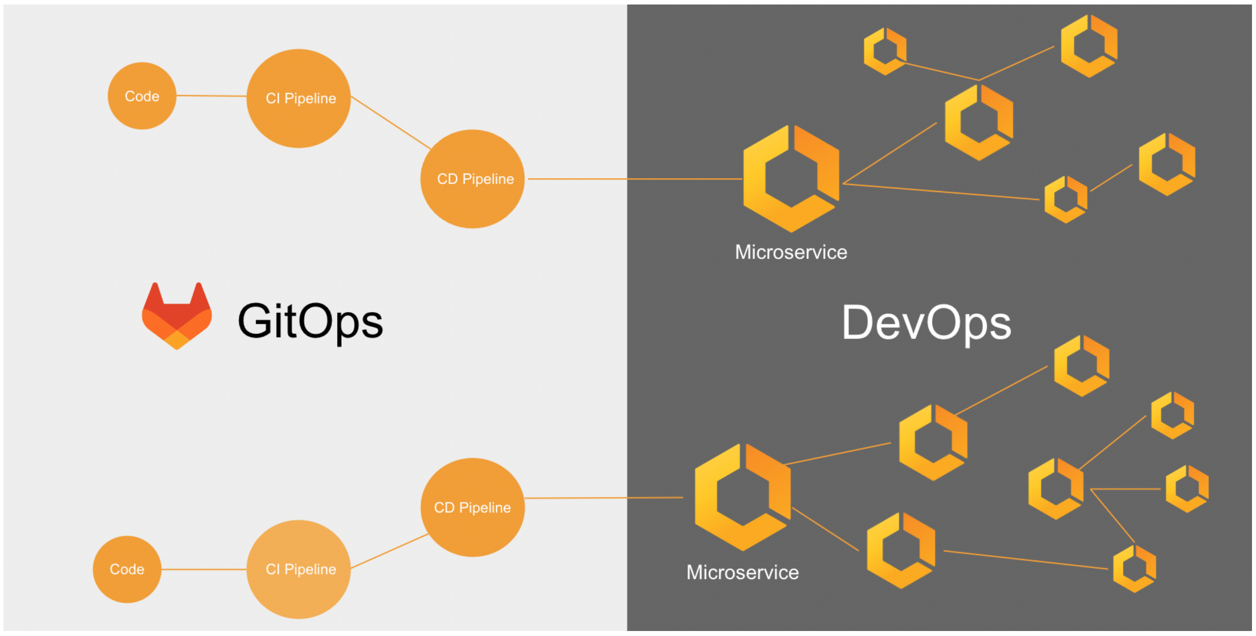
This paper provides a look at CtrlStack, which allows DevOps teams to visualize all data silos as one connected graph, bringing a better representation of reality. This visualization can be used to improve observability, simplify actions, and then automate troubleshooting workflows to speed:

- **Symptoms** - What went wrong in the system?
- **Diagnosis** - What changed? What action can be taken?
- **Resolution** - What needs to be done?
- **Reporting** - What needs to be learned?

When it comes to DevOps, there are several factors converging that are forcing enterprises to rethink observability and incident response:

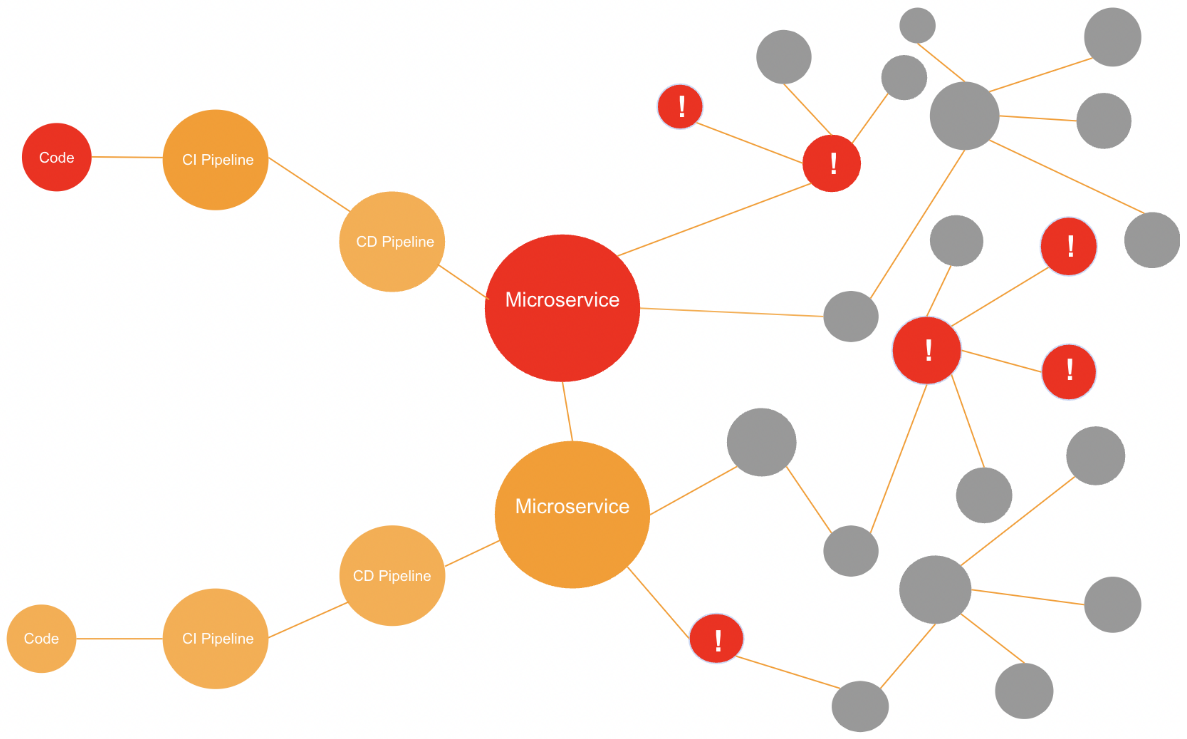
01

DevOps teams are faced with growing business pressure to improve service reliability and uptime as enterprises increasingly rely on microservice architectures and CI/CD to enable continuous delivery of new features and fast-evolving applications. There are tools to handle microservices and CI/CD pipelines in isolation, but they are effectively connected. Harnessing the power of those connections allows teams to understand critical data relationships, locality and distance, so when issues occur teams can focus on nearby connections.



02

With the scale and complexity of distributed systems with microservices, cloud computing solutions, and continuous deployment, an incident could trigger numerous alerts across services within the tech stack. To accurately diagnose root causes that may be several steps away from the initial observed anomalous service, RCA must be efficient as you scale to ingest larger volumes of data.



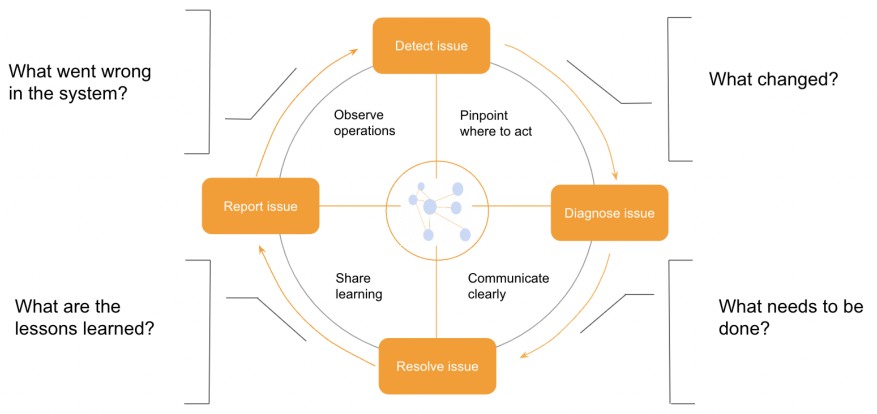
03

Traditional monitoring tools that focus on collecting data types - logs, metrics, and distributed traces - is a siloed approach to observability. Simply collecting these data types doesn't provide faster or better problem resolution. Many companies still struggle with knowing what data is important and what's not when it comes to real-time troubleshooting. When we unify disconnected data, it becomes easier to visualize the state of a system and how the pieces interconnect and interact.

04

75% of performance problems can be traced back to changes in the environment; 67% of organizations can't identify the relevant change. Changes are made to the code or the cloud infrastructure, either directly or through the infrastructure as code (IaC) templates, for a variety of reasons. The risk of misconfiguration and performance issues increases when these operational activities are not continuously monitored and correlated across the stack.

What if we could automatically construct the dependency graph, connect the operational data (performance metrics, logs, and operational activities), and model the causal links between the data for real-time root-cause diagnosis?



05

The knowledge gap between senior engineers and less experienced engineers gets larger over time and makes real-time troubleshooting more challenging. Documentation has a half-life measured in weeks or months depending on release cadence. Less experienced engineers often don't know what data is important to a specific incident and lack the ability to understand the links between cause and effect. Most companies which deploy daily report that their engineers spend at least half of their time troubleshooting and debugging.

So where does this leave us?

Organizations are facing tough problems - from needing to gain meaningful insights out of massive volumes of data to having visibility across vast systems and the ability to fix issues quickly to avoid downtime. Many DevOps teams are stuck with legacy monitoring tools that collect disconnected data, and are tasked to find solutions that improve service reliability and uptime without disrupting their workflow. This has led to an influx of vendors trying to reconnect the fragmented data after the fact, using fragile machine learning techniques. Relying on machine learning before building that connection/knowledge foundation does not allow vendors to take full advantage of machine learning. Quality data as well as missing relevant data, such as change events, are directly proportional to the performance of a machine learning model.

On the other hand, having a DevOps graph that links all the infrastructure and microservices together lets teams see hidden relationships and find correlations quickly. In this paper, you'll learn how CtrlStack harnesses those connections to efficiently connect cause and effect to speed symptoms, diagnosis, resolution, and reporting for real-time troubleshooting. This means that teams can do their investigation and get to the root cause within a few minutes, in just one click.

Leveraging a Graph for Incident Response

The traditional method of looking at metrics, searching through logs, line by line, and attempting to pivot off of the data, can take several hours to determine the causes and explain how the event occurred. There are hidden relationships in all the data that can only be expressed through an engineer's knowledge graph.

When we pull out the relationships from the data to form a real-time, dynamic graph, the effects (symptoms) and related causes are more obvious. We want to move from observing individual data points and then slowly connecting them in our heads, to observing all the data points and connections in the same context. In doing so, our process of identifying and resolving issues is faster and more accurate.

In modern IT environments, different teams are constantly integrating new features, new technologies, and renewing their stack. When changes are made to improve application performance or to harden systems, they often introduce new pathways to failures. Although nothing replaces earned experience and expertise, tools that help to improve an engineer's ability to pivot and diagnose issues would significantly cut the problem resolution time.

Use Case: Service Fault at Google

The following use case from Google illustrates how incident response works in practice. This example shows what happens when a team of experts tries to debug a system with many interactions, and no single person can grasp all the details. This postmortem report shows how an incident with a service fault at Google could have been managed better with a DevOps graph. Even with several teams of experts, the GKE CreateCluster outage took 6 hours and 40 minutes to fix.

Context

Google Kubernetes Engine, or GKE, is a Google-managed system that creates, hosts, and runs Kubernetes clusters for users. This hosted version operates the control plane, while users upload and manage workloads in the way that suits them best.

When a user first creates a new cluster, GKE fetches and initializes the Docker images their cluster requires. Ideally, these components are fetched and built internally so we can validate them. But because Kubernetes is an open source system, new dependencies sometimes slip in through the cracks.

Incident

An on-call SRE for GKE declared an incident when she verified CreateCluster probe failures were occurring across several zones; no new clusters were being successfully created.

Here's the timeline of the incident:

- **7 a.m. (Assessed impact).** On-call SRE confirmed users were affected by the outage.
- **9:10 a.m.** So far, the incident responders knew the following:
    - Cluster creation failed where nodes attempted to register with the master.
    - The error message in cluster startup logs indicated the certificate signing module as the culprit.
    - All cluster creation in Europe was failing; all other continents were fine.
    - No other GCP services in Europe were seeing network or quota problems.
- **9:56 a.m. (Found possible cause).** 2 team members identified a rogue image from DockerHub
- **10:59 a.m. (Bespoke mitigation).** Several team members worked on rebuilding binaries to push a new configuration that would fetch images from a different location.
- **11:59 a.m. (Found root-cause and fixed the issue).** SRE on-call disabled GCR caching and purged a corrupt image from the European storage layer.

Google's Diagnosis Approach

The team had several documented escalation paths which helped to quickly get domain experts to get onboard. Logging was insufficient for diagnosis. In fact, digging into logs and finding the error message at the start distracted the team. The corruption on DockerHub was not the real issue. A handful of first responders pursued their own investigations without context and coordination. Without seeing the DevOps graph that tracks all the system changes through a single lens, a lot of time and resources were wasted jumping between tools and tabs.

A Better Approach: Incident Diagnosis with a DevOps Graph

In this case, the responders would have benefitted from a platform that connects cause to effect to find the root cause quickly. This platform would present the DevOps graph in a topology view that shows the connections/relationships between the entities such as the Docker image running in a Kubernetes cluster that's pulled from DockerHub, or in this case, Google Container Registry (GCR). You would also see where in the timeline did the Docker configuration change and what changes were made. Being able to see the configuration changes and the relationships between the entities, in relation to the incident, in one place would have expedited the diagnosis and resolution of the incident.

With a platform like CtrlStack, all the actions taken by team members in the system are automatically captured. This ensures anyone jumping onboard will have the right context, can see what has been done, when, and by who. Nothing will be lost in translation. Most importantly, teams don't have to spend more time writing an epic post-mortem after having spent almost 7 hours resolving the issue.

Conclusion

A DevOps platform that connects all data, including change data, and relationships can significantly enhance teams' troubleshooting experience. When information is provided in context, and incident triage can be largely automated through a connected graph, teams can troubleshoot in real time.

With CtrlStack, moving beyond legacy solutions no longer needs to be an intimidating idea. CtrlStack uses a native graph approach to represent the "data between the data" - the relationships between metrics, events, logs, and traces. This means teams can continue to use their existing operational data without the mental burden of tracking how infrastructure and services interconnect, and what changes have been made in their environment. Teams get the added benefit of seeing the impact of a change - code, configuration or action - and quickly trace backward to get to the root cause of the problem.

Learn more about how CtrlStack helps troubleshoot cloud applications in real time here.

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.