OpsCruise

Autonomous Operations for Cloud Applications

Technical e-Book

The Modern Application Environment

Since the introduction of the concept in 2013, containers have become the buzz of the IT world. It's easy to see why: application container technology is revolutionizing app development, bringing previously unimagined flexibility and efficiency to the development process. Businesses are embracing containers in droves. According to Gartner, more than half of global enterprises will be running containerized applications in production by 2020, up from less than 20 percent today. And IDC predicts by 2021, more than 95 percent of new microservices will be deployed in containers. That mass adoption makes it clear that organizations need to adopt a container-based development approach to stay competitive.

And to help manage those containers for rapid continuous integration, deployment and high availability, organizations are embracing orchestration platforms such as Kubernetes. Kubernetes is one of the most popular open source projects today with on-prem distributions available from many vendors and hosted/managed options available from the top cloud providers.

Monitoring and Observability Challenges

For all the business agility that microservices, containers and orchestration bring to IT organizations, they can also make assuring application performance even more complex. Some of the challenges they present include:

- Many layers of abstraction—These environment have multiple complex layers of dependencies for your application, including application middleware (e.g. relational and NoSQL DBs, messaging, APIs, etc.), the orchestration platform and infrastructure (compute, network, storage, etc.) on-prem or in the cloud. Further, many apps have dependencies on 3rd party SaaS and APIs over which they have no control.
- Significant Blind Spots—Containers are designed to be disposable. Because of this, they introduce several layers of abstraction between the application and the underlying hardware to ensure portability and scalability. This all contributes to a significant blind spot when it comes to conventional monitoring.
- Increased Need to Record—The easy portability of so many interdependent components creates an increased need to maintain telemetry data to ensure observability into the performance and reliability of the application, container and orchestration platform.
- The Importance of Visualization—The scale and complexity introduced by containers and container orchestration requires the ability to both visualize the environment to gain immediate insight into your

infrastructure health but also be able to zoom in and view the health and performance of containers, node and pods. The right monitoring solution should provide this workflow.

- Don't Leave DevOps in the Dark—Containers can be scaled and modified with lightning speed. This accelerated deployment pace makes it more challenging for DevOps teams to track how application performance is impacted across deployments.

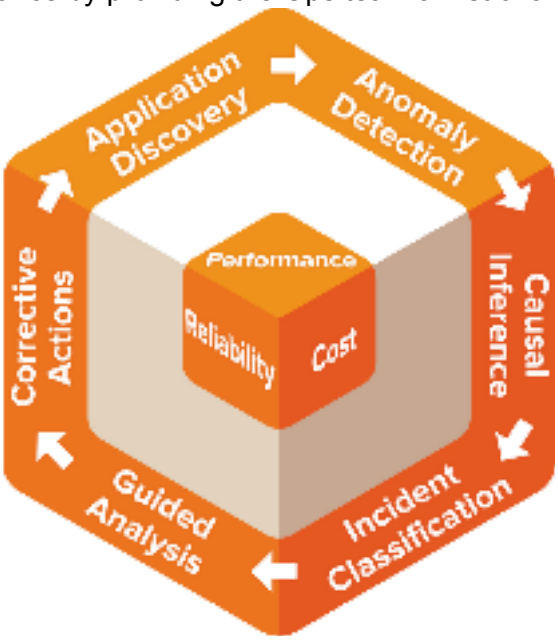
The good news is that many of these environments and systems are much more observable. Container and orchestration layers expose data (e.g. cAdvisor for containers and Kube State Metrics and etcd metrics for Kubernetes, etc.). Further, public and private clouds provide significant event & metric data for all of their resources via centralized platforms (e.g., AWS Cloudwatch, Azure Monitor, vCenter, etc.). Finally, application middleware developed in the open source community (e.g., MongoDB, Cassandra, Elastic, Kafka, NGINX, etc.) are well instrumented with hundreds of resources, configurations, state metrics as well as logs.

What's more, there are numerous next-gen open source and commercial tools that do a good job of aggregating metrics, logs and traces. However, they often only cover one specific layer/domain and leave it to developers and sys admins in operations/SRE teams to make sense of the thousands of different signals arriving every second.

A good container monitoring solution with contextualization will enable you to predict performance issues before they impact users and pin-point root-causes with fine-grain recommendations/remediations at all layers, including 3rd party SaaS services. It will provide L1 and L2 operations/SRE staff actionable insights as opposed to thousands of metrics and hundreds of pretty charts.

OpsCruise Autonomous Operations Overview

OpsCruise addresses the challenge of managing application performance by providing the Ops team a first of a kind autonomous dynamic performance assurance platform. The solution comprises two parts:



OpsCruise Live App Map™ for Actionable Insights provides automated granular visibility into where and why performance problems are occurring, enabling direct pinpoint intervention

OpsCruise Assure™ for Prescriptive Tuning restores application service levels with automation, assisting the decision and control process

The platform consumes existing monitoring, event and configuration data to automatically build a real-time application-wide model using a unique patent-pending application-centric approach. This model is built using curated application knowledge augmented by multiple machine learning techniques. It provides a deep granular understanding of the application behavior of all underlying components and their interdependencies across the application and down to the infrastructure level. Using this model in a predictive

mode, OpsCruise provides continuous application performance insights – detecting early onset of problems, isolating the cause and locality of the problem and recommending corrective actions.

The resulting benefits positively impact both top and bottom line: increased uptime through proactive problem resolution to meet performance SLAs of customer facing applications and services; greater organizational agility from improved efficiency and predictability of their DevOps process for new applications; and reduced cost of infrastructure resources as well as decreased dependence on highly skilled operators.

Finally, OpsCruise is low friction. It is delivered as-a-service, requires no agents to deploy, no changes to your application code and integrates with popular monitoring and service management tools.

Technology and Architecture

Guiding Principles

OpsCruise's SaaS is designed from the ground up to provide Operations (Ops) or SRE teams an actionable platform to continuously manage their containerized applications per service levels. The approach is novel in that it creates a model-driven control system that embeds curated knowledge to provide more context and fill in the missing information using appropriate machine learning (ML) techniques. OpsCruise builds on the following four guiding principles to design and build its platform:

1. Use model-based control for predictive and autonomous Operations
2. Leverage curated knowledge related to the application and the infrastructure to create white or gray box models of the application and its components
3. Decompose the overall problem of automated application management into multiple stages from auto-discovery and auto-build of the application structure to executable remediation actions
4. Supplement gaps in understanding of application behavior with appropriate ML techniques

OpsCruise uses multiple open sources of data and accessible 3rd party sources for metrics, events, configurations, as well as operator input.

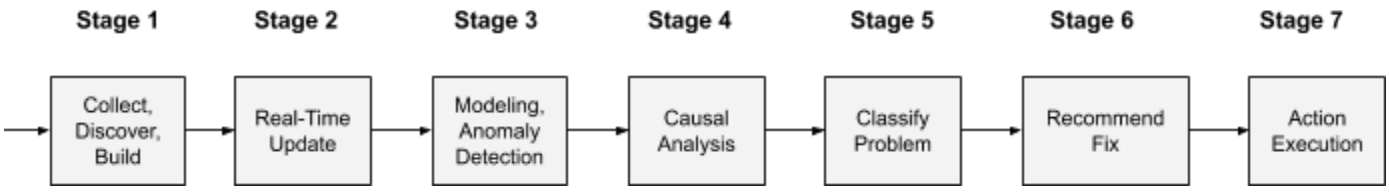


Figure 1. Pipelined analytical processing of data in multiple stages

Deployment Model

OpsCruise's deployment model depicted in Figure 2 is predicated on providing Ops teams in-depth real-time visibility into and control without assuming knowledge of the application code. Most importantly, OpsCruise's leverages the application owner's existing open monitoring framework that is compatible with the Kubernetes (K8s) stack, viz., K8s, Prometheus, Istio, Jaeger, etc..

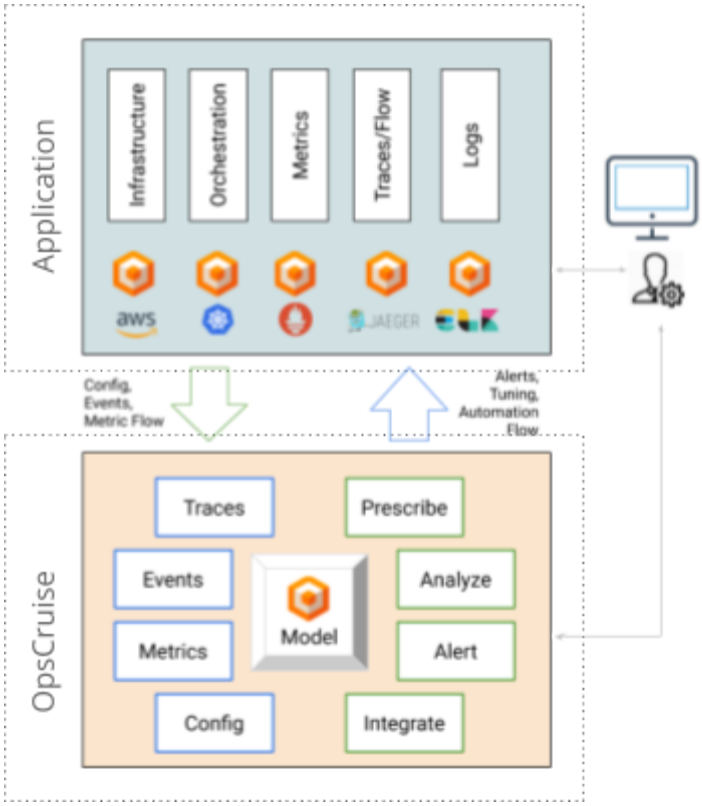


Figure 2. OpsCruise's deployment model

1. OpsCruise is deployed in two parts with data collection gateways, the south side, installed behind existing monitoring framework of the application, and the SaaS controller, the north side, in the cloud. The deployment can be in any of DevOps to Production environments depending on the application owner's cloud.
2. OpsCruise works with application containers managed by K8s and where all metrics related to the containers and services are collected using Prometheus. Other open source monitoring components in the K8s stack will also be used including tracing from Jaeger, service mesh data from Istio, as well as 3rd party monitored metrics where available.
3. Data Gateways on the application monitoring plane pull data and send to the north side comprises the following:
    3.1. Container, application and custom metrics
    3.2. Kubernetes state metrics and events
    3.3. Cloud infrastructure metrics

3.4. Cloud service related metrics
3.5. Network Flow data at Layer 4 and Layer 7 between containers and nodes

All gateways and the node exporter are deployed as containers or pods in the existing monitoring plane. The gateways are open source so the customer can review and control the information forwarded to the controller.

Multi-stage Pipelined Analysis

OpsCruise recognizes that microservices applications are complex, heterogeneous, and dynamic distributed systems, and that to create a model-based control system requires multi-step sequence of processing where each step or stage poses a different problem statement requiring a different solution as shown in Figure 3.

The multi-stage analytics and processing are deployed in pipeline mode, beginning from the initial processing of the data sent by the data gateways from the application environment in Stage 1 to the execution of any necessary remediation action in Stage 7. The objective of each stage is to gain progressively increasing visibility into the internal state of the application across its components so that we can detect, analyse, and isolate an incident and its cause and then determine the required corrective action.

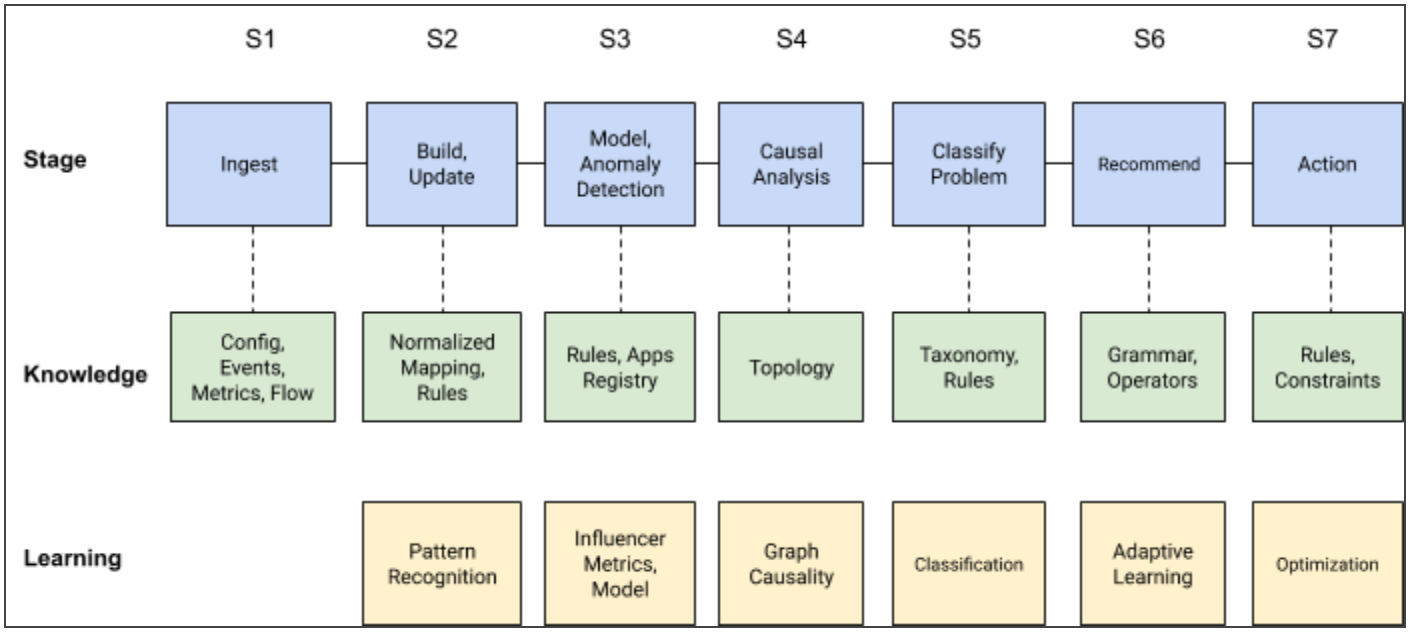


Figure 3. Contextual processing across OpsCruise stages

The processing done in each stage of the pipeline of Figure 1 uses a combination of curated knowledge specific to the stage and the appropriate machine learning (ML) techniques as needed for the objectives for that stage as shown in Figure 3.

Stage 1 (S1): Ingest Data and Build Application Graph

S1 maps all data collected from the south side into normalized form to build a multi-layer graph Application Graph (App Graph) from configuration, metrics and network data. The App Graph represents the structural topology and directional dependencies across all three layers: the application layer, the orchestration (Kubernetes) layer, and the infrastructure (cloud) layer.

Stage 2 (S2): Real-Time Update of Application Graph

Stage S2 is used to enrich and update the App Graph with the most recent metrics and event data for each entity across all three layers. This provides the most recent known state of the application and its infrastructure.

Stage 3 (S3): Build Predictive Behavior Models and Detect Anomalies

Stage S3 builds a predictive understanding of the complete application using a priori curated knowledge as well as a multiplicity of ML techniques.

The model fingerprints the application behavior, and predicts expected values of key indicators, 'influencer metrics', or the performance metrics based on the demand. The influencer metrics are primary indicators of the health of the applications and are shown in the App Map, relieving the Ops team from guessing which metrics to track.

Known Application Registry: because most applications use a number of common known components, e.g., Kafka or ElasticSearch, OpsCruise uses a 'factory' process to model these known components to create offline behavior model templates for them. The model template is then updated in runtime to reflect the behavior in the operating environment.

Additionally, after an anomaly is detected, the application model is used in an interpretive mode to determine which specific influencer metrics might have caused the anomaly.

Stage 4 (S4) - Causal Analysis

S4 is launched whenever any anomaly is detected in S3, from which it receives all relevant labels and metadata related to the anomaly and the conditions leading to it. OpsCruise's causal analysis is based on a deterministic approach and not on agnostic correlation, and relies on multiple contextual pieces of information.

Using the above analysis allows us to eliminate false positives, reduce false correlation, significantly reduce the computational overhead, and uncover hard to detect long chain dependencies.

Stage 5 (S5): Problem Classification

The goal of S5 goal is to classify the anomaly and causal analysis information received from S3 and S4 into an existing taxonomy that has sufficient detail on the nature of the problem and suggested remediation actions.

The classification also uses curated knowledge of typical operations problems and conditions as they relate to failures and incidents that are known to occur in containers as well as known applications. As more varieties of

anomalies with different attributes and environmental conditions are added, the problem classification is adapted to address a greater breadth of problem types.

Stage 6 (S6): Recommended Remediation

S6 provides recommended actions and is tightly coupled to the problem classification of Stage 5. A problem is described in terms of details on the anomaly entity and its operating environment, and is associated with a related remediation action that may make necessary changes to the configurations, resources, or services. The recommendation is forwarded to the Ops team via the incident management system, such Pager Duty or ServiceNow. Once the action has been executed in Stage 7, and the results are noted in the incident management system, the efficacy of the recommendation is recorded. A closed loop feedback action is taken to improve the recommendation.

Stage 7 (S7) - Action

S7 is the actual execution of the recommended action for remediation. Because actions to be taken are heavily dependent on the organization's process and policies, the action has to be mapped to the organizations' operations runbook. Further, given the many constraints, the action execution will be implemented as an optimization solution. OpsCruise expects most organizations will manage and implement their own action plan.

Functional and Operational Aspects

There are a number of key functional elements that OpsCruises platform addresses that are noteworthy. These include extensibility of the solution and use in capacity planning.

Extensibility

OpsCruise's solution applies to general microservices applications and is not restricted to applications that are comprised only of containers, but include both SaaS and serverless components. Containerized application models have more explicit visibility and control over their orchestration and infrastructure metrics. Their models include configuration settings and allocation of resources at both levels, and thus provide a more granular control on their performance. OpsCruise can also model the performance of non-orchestrated components but because SaaS and serverless components are not orchestrated, there are fewer choices in managing their configurations, e.g., memory allocated and timeout specified in an AWS Lambda function, and therefore expose less granular control.

Capacity Planning for Scale

While the primary use of OpsCruise is for autonomous operations of cloud applications in production, the application model can provide first-order insights into the required infrastructure as the application is scaled. The basis for the planning is estimating the resource and service demands from the behavior model of all application components.

Ease of Deployment

OpsCruise deployment of the data collection is designed to have minimal impact on the existing operations environment. It creates no application impact since the gateways are deployed as pods in the monitoring plane with

a simple container installation process consistent with that used by Ops to deploy other containers. The collection gateway software are simple open Go code that Ops can inspect. The gateways pull data from existing Prometheus servers, Kubernetes and cloud monitoring APIs.

Managing Security

The data collected by the gateways are operational metrics that do not contain payload that may have PCI or PHI concerns. No secrets are transmitted to the SaaS. In addition, the Ops team has full control over metrics or names that they feel are sensitive and can anonymize or remove by setting filters.

Cloud Agnostic

OpsCruise SaaS is cloud agnostic and therefore does not depend on whether the organization is running on public cloud or on-premise. The only requirement is that the application containers are orchestrated by Kubernetes.

Use Case Examples

We provide two use cases where OpsCruise can provide resolution but traditional approaches used by Ops would have difficulty addressing

e-Commerce transaction rate dropping

- Operations Issue: e-commerce transaction rates dropping unexpectedly
    - Detects e-commerce transaction times have increased
    - Detects backend database MySQL writes are taking longer
    - Concludes incorrectly that MySQL is the source of the problem
- OpsCruise solution
    - Detects e-commerce transaction times have increased
    - Detects backend database MySQL writes are taking longer
    - Notes that incoming transaction volume (demand) has not changed
    - Recognizes that the IO workload to MySQL has changed significantly: # writes to disk drastically increased, compared to earlier situations for the same transaction volume
    - Existing disk IOPs cannot handle the total IOPs given the increased writes to MySQL
    - Notifies Ops that nature of transactions have changed given existing workflow: recommends
        - validate that the new transactions types are allowed
        - and if so, increase disk IOPs to handle the changed load

Multiple concurrent alerts with many false alerts

- Operations Issue: streaming IoT application uses Kafka, Storm, and Elasticsearch. Kafka queues are building up and lag is increasing
    - Detects Kafka consumer lags are increasing, and alert is created due to the SLO breach
    - Storm, the component downstream from Kafka, has no alerts or anomalies
    - Elasticsearch, downstream from Storm, shows CPU saturation, and alert is created
    - No clear reason for these alerts to be related

- OpsCruise solution
    - Detects that Kafka consumer lag is increasing
    - Causal analysis initiated from Kafka using dependency chains from known topology
    - Kafka model shows its behavior is as expected, and no services it depends on in K8s or infrastructure level dependencies show no issues
    - Conclude Kafka is not the source of the problem
    - Follow chain to Storm and use model to check and verify Storm is not a source of the problem
    - Follows chain from to Elasticsearch and detect existing alert due to CPU saturation
    - Link Kafka alert ticket with that of Elasticsearch

Summary - What makes OpsCruise Unique

OpsCruise provides a break-through approach to performance management and autonomous operations for cloud applications. OpsCruise's platform has the following advantages over traditional monitoring tools:

- Cohesive and Integrated Application Observability: OpsCruise builds real-time full-stack granular observability into the application, called Application Map, that comprises all layers of the application and their interdependencies across the application and down to the infrastructure level.

- Live Tracing: lightweight real-time flow tracing for operations team: OpsCruise provides granular layer 4 and 7 performance analysis across all service components for causal analysis. It can provide complete real-time coverage and complement distributed tracing or service mesh technologies that require code tagging and significant infrastructure if running continuously.

- Open Source Monitoring: OpsCruise is not a monitoring tool but uses existing monitoring frameworks and therefore does not require installing any application level agents or software.

- Automated Key ('Influencer") Metrics Selection: OpsCruise automatically determines the key influencer metrics that matter for application,component. This enables Ops to view only the metrics that are relevant for understanding the application, without guessing which metrics are important.

- Predictive Insights: OpsCruise's detailed understanding and use of the behavior model provides continuous application performance insights – from detecting early onset of problems to recommending corrective actions, instead of relying on alerts where the only options are recovery from failures.

- Action Recommendation: OpsCruise checks whether the application is behaving correctly, and on detection of a problem, analyses and classifies the problem down to the entity that requires remediation, and recommends corrective actions.

About OpsCruise

OpsCruise is a start-up venture funded by The Fabric and based in Sunnyvale, CA. Our founders are industry veterans and innovators with expertise in IT operations, data center technologies, computational analytics and machine learning.