CloudFabrix

WHITEPAPER

Advancing the **Data-Centric AIOps with Composable Analytics**

Table of Contents

Executive Summary

Modern IT environments are witnessing unending complexity, with ever increasing amount of hybrid workloads, numerous siloed operational tools and noisy data - all resulting in increased operational burden, excessive worker toil, missed critical incidents and lost opportunities.

The only way to effectively address this unending IT complexity is by utilizing Artificial Intelligence for IT Operations or AIOps, to gain predictive intelligence and drive towards autonomous operations. Gartner, says 40% of I&O teams will adopt AI-augmented automation to boost IT efficiency by 2023; and IDC predicts global spending on AI will reach $500 billion in 2024.

As AIOps solutions take on the monumental challenge of processing vast amounts of data from cross-domain environments, following needs become imminent for AIOps to solve and address:

- Serve multiple stakeholders (ex: executives, operators, administrators, experience desk, service desk, engineering, etc.)
- Provide curated services for multiple teams (ex: ITOps, ITSM, NOC, ServiceDesk, DevOps, CloudOps, etc.)
- Ingest and process vast amounts of data from hybrid environments (ex: edge, branch, data center, multi-cloud, etc.)

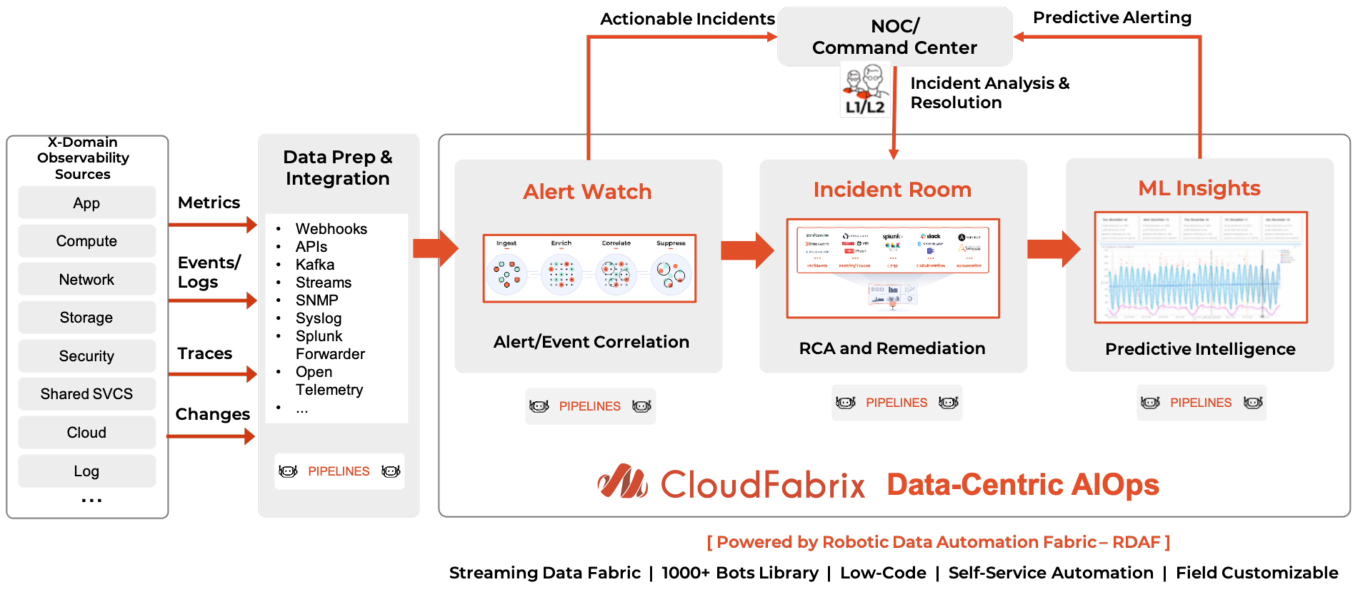
This is a tall order, and only a modern AIOps solution that is data-centric & has composability as a core architectural capability can meet such varying needs.

This whitepaper explores how CloudFabrix Data-Centric AIOps solution addresses the above needs with the following 3 composability pillars:

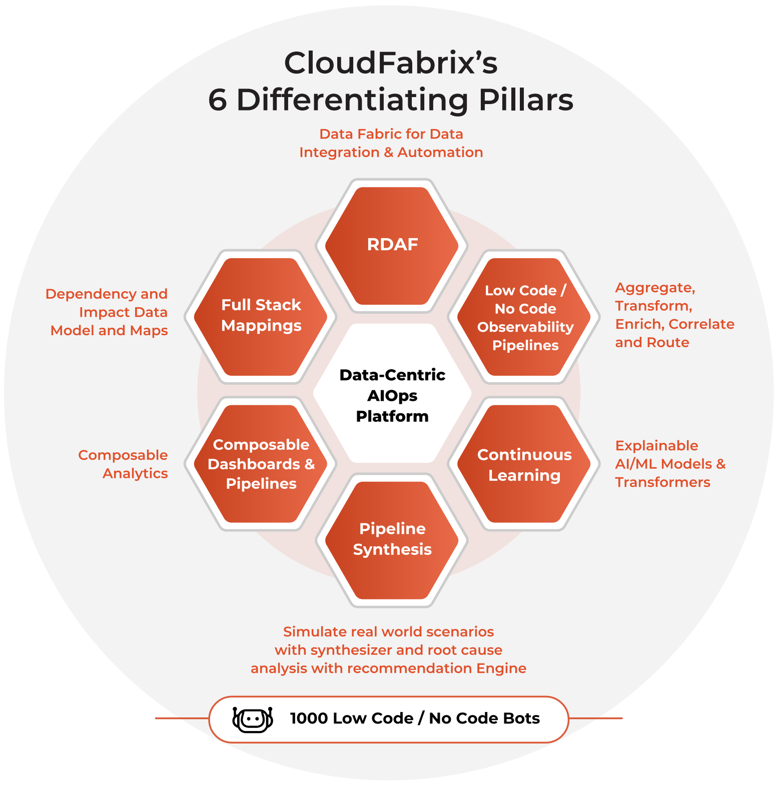
- Composable Dashboards and Analytics
- Composable Services
- Composable Pipelines (with low-code/no-code)

CloudFabrix Data-Centric AIOps: At a Glance

CloudFabrix AIOps is a leading cross-domain AIOps solution that uses AI/ML technologies to reduce noisy incidents, accelerate incident resolution with root cause analysis and provide predictive intelligence to prevent unplanned outages or service degradations. Customers realize benefits by cutting down unactionable incidents by 90% or more, incident resolution times cut down from hours to minutes and minimal unplanned service outages - all resulting in flawless customer experiences, increased IT productivity and reduced operational expenses.

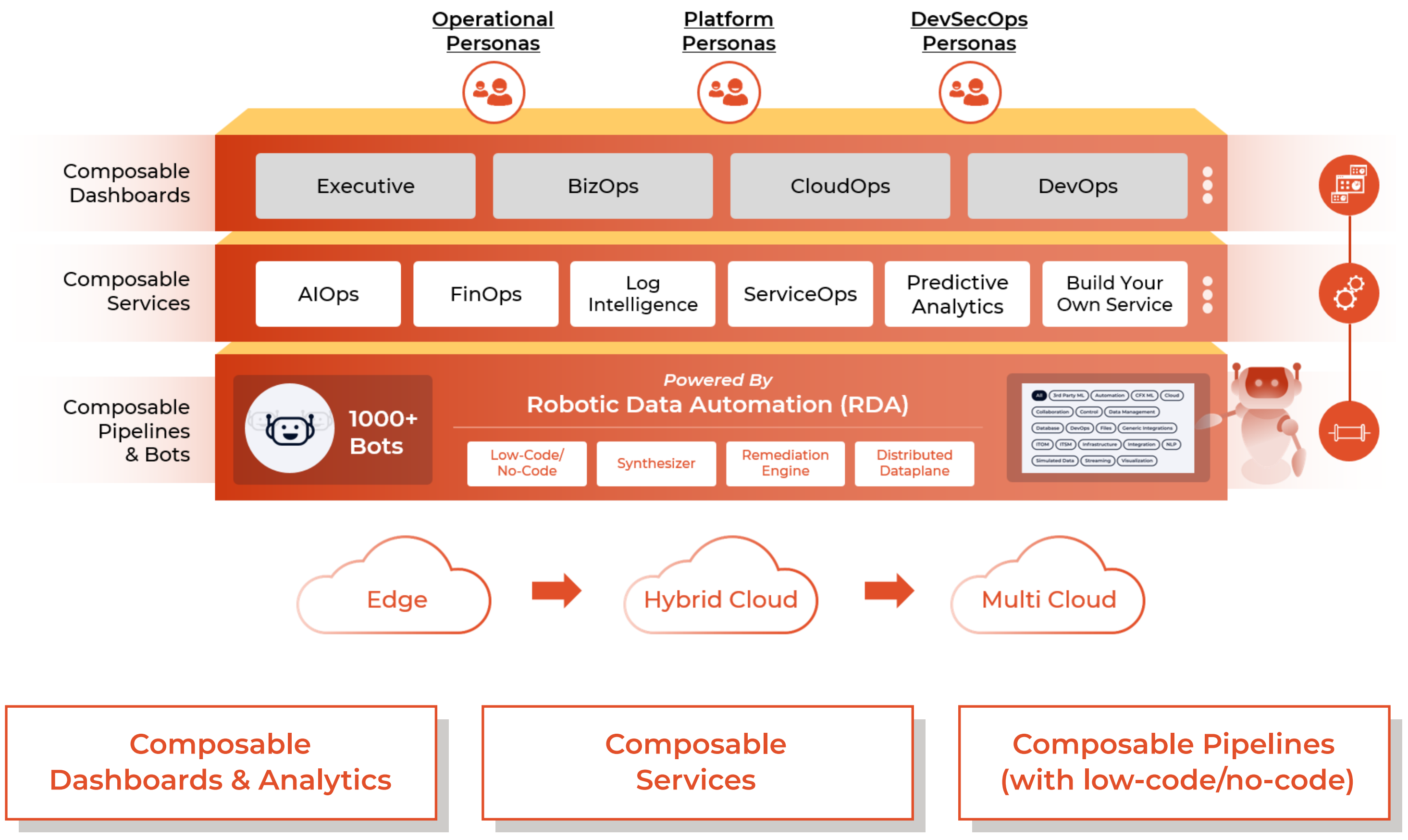


- **Data Fabric:** Freedom to ingest and integrate data from any source, any tool,any environment
- **Low-Code/No-Code Observability Pipelines:** Limitless services and data automations to run alongside AIOps
- **Composable Dashboards, Services, Pipelines:** To address use cases and needs of various stakeholders
- **Full Stack Intelligence:** Establish full stack application dependency map using agentless multi-protocols
- **Continuous Learning:** Learn from data with explainable AI/ML models
- **Deployment Choice:** Freedom to deploy on-prem, public cloud or consume as a service (SaaS)


CloudFabrix's
6 Differentiating Pillars
Data Fabric for Data Integration & Automation
RDAF
Dependency and Impact Data Model and Maps
Full Stack Mappings
Low Code / No Code Observability Pipelines
Aggregate, Transform, Enrich, Correlate and Route
Data-Centric AIOps Platform
Composable Analytics
Composable Dashboards & Pipelines
Continuous Learning
Explainable AI/ML Models & Transformers
Pipeline Synthesis
Simulate real world scenarios with synthesizer and root cause analysis with recommendation Engine
1000 Low Code / No Code Bots

Composability Pillars of Data-Centric AIOps

A good AIOps solution needs to serve multiple stakeholder teams, address multiple use cases, ingest data and integrate with various tools from distributed environments. Legacy client server architectures, or shrink-wrapped softwares are not capable of serving the AIOps needs of modern enterprises. The AIOps solution needs to be 100% data-centric, built with modern architectural principles and help achieve technical/business outcomes that various stakeholders pursue. CloudFabrix solution is built with composability as a core architectural capability, and the incarnation of composability is made available via three value pillars depicted in the following illustration diagram



A foundational component of the AIOps solution is CloudFabrix Robotic Data Automation Fabric (RDAF™), which helps in automating various data integration, enrichment and operational workflows that help accelerate AIOps implementations. Following sections of the document will explore in detail each of the composability pillars of the AIOps solution

I. Composable Dashboards

AIOps has multiple stakeholders, with each stakeholder requiring different data, insights and tools to perform his/her intended interaction and achieve desired outcome in the least amount of time and with most pleasant user experience.

- Highlights: Easily build persona-based dashboards that provide key data, insights and tools to achieve intended technical/business outcome
- Benefits: Improved user/operator experience, improved productivity and agility
- How it Works: Use out of the box persona-based dashboards, customize them, or compose your own - using visual drag & drop tools or JSON based definitions

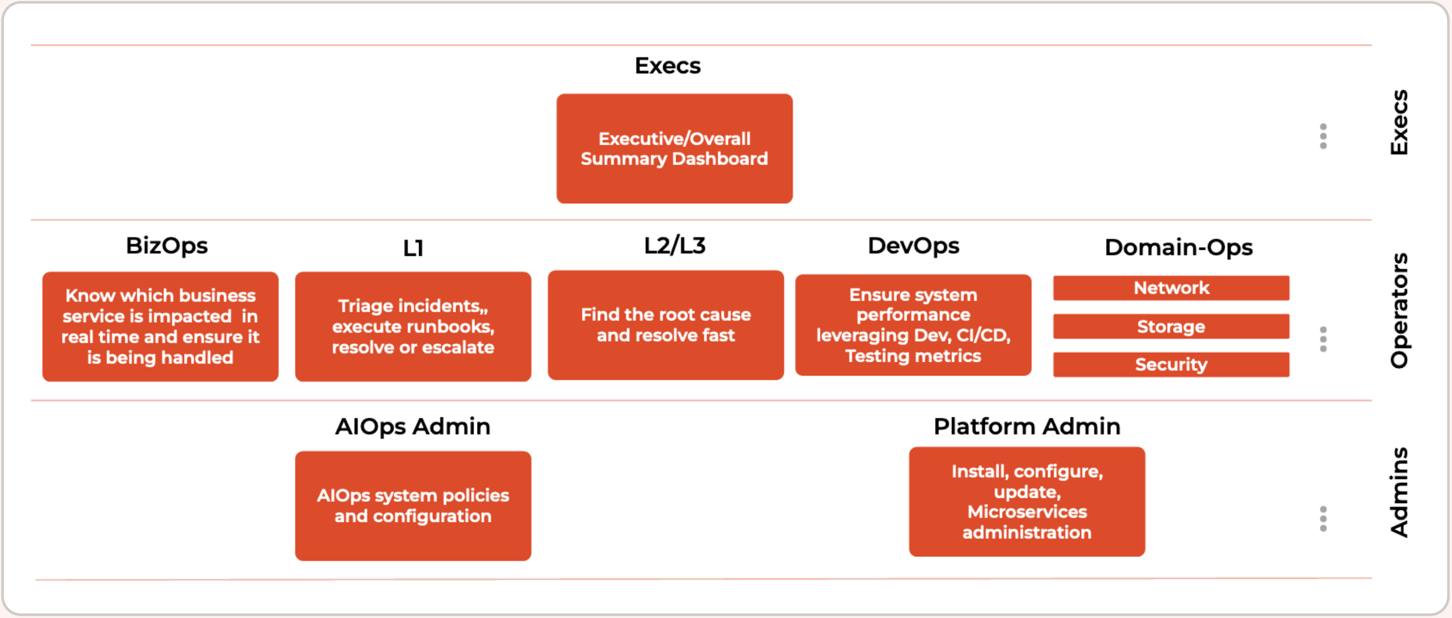

Execs
Executive/Overall Summary Dashboard
Execs
BizOps
Know which business service is impacted in real time and ensure it is being handled
L1
Triage incidents,, execute runbooks, resolve or escalate
L2/L3
Find the root cause and resolve fast
DevOps
Ensure system performance leveraging Dev, CI/CD, Testing metrics
Domain-Ops
Network
Storage
Security
Operators
AIOps Admin
AIOps system policies and configuration
Platform Admin
Install, configure, update, Microservices administration
Admins

Fig: Various stakeholders and their intended interaction with AIOps platform.

CloudFabrix AIOps provides following dashboards out-of-the-box, and each one of these dashboards can also be customized or new ones can be created.

01. Executive Dashboard

Provides overall summary, key success metrics, AIOps performance trend, incidents trend and efficiency metrics.

- Open P1 and P2 Incident count, trend
- Noise Reduction efficiency and trend
- Auto Remediation efficiency (system resolved incidents vs agent resolved incidents)
- MTTR trend
- Alerts/Events Heatmap
- Top Noisy Sources

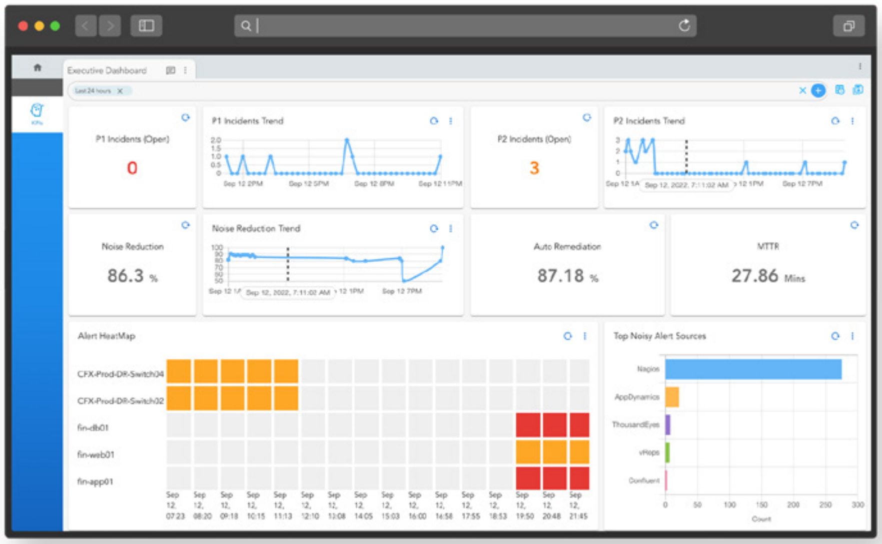
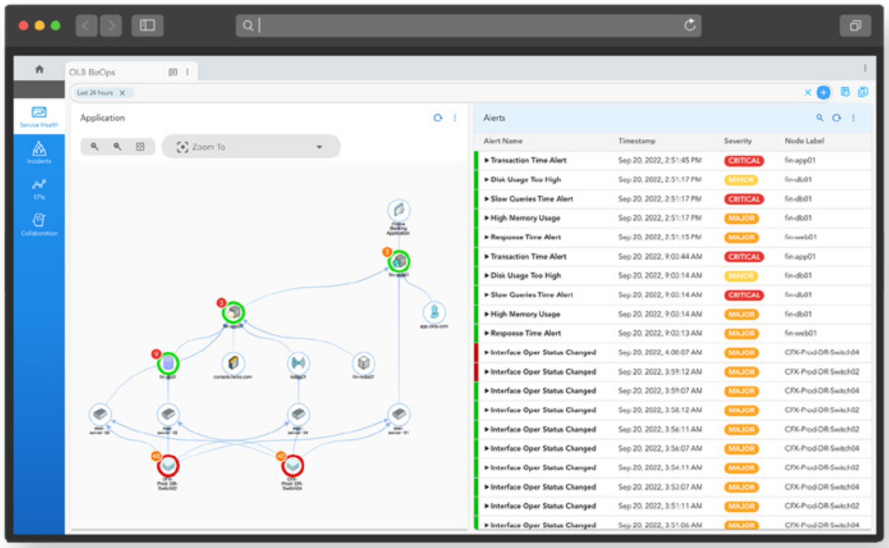


Fig: Example of Executive Dashboard

02. BizOps or Experience Desk Dashboard

<u>**Responsibility:**</u> Ensure highest customer experience and satisfaction. Ensure that services are healthy, and if any issues are observed, ensure that incidents or trouble tickets are created and service desk is looking at the issue.

- <u>Service Health:</u> Visually identify service health and quickly spot any operational issues
- <u>Incidents:</u> Get a view of incidents to ensure operational issues are being tracked
- <u>KPIs:</u> View of key operational metrics of business or IT components

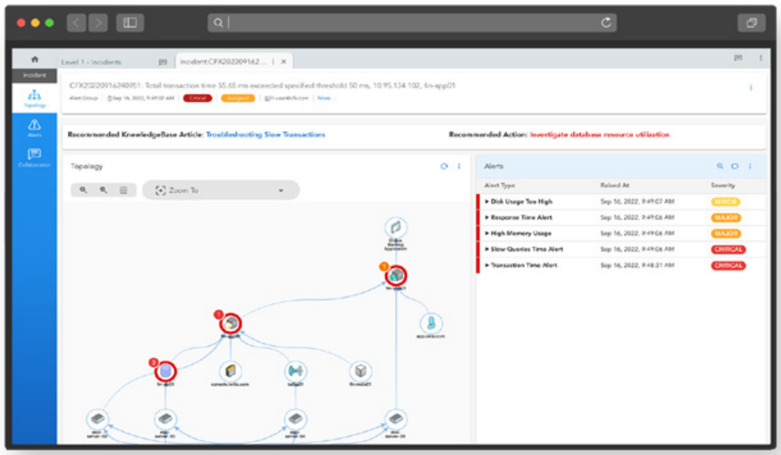


Alert Name	Timestamp	Severity	Node Label
▸ Transaction Time Alert	Sep 20, 2022, 2:51:45 PM	CRITICAL	fin-app01
▸ Disk Usage Too High	Sep 20, 2022, 2:51:17 PM	MINOR	fin-db01
▸ Slow Queries Time Alert	Sep 20, 2022, 2:51:17 PM	CRITICAL	fin-db01
▸ High Memory Usage	Sep 20, 2022, 2:51:17 PM	MAJOR	fin-db01
▸ Response Time Alert	Sep 20, 2022, 2:51:15 PM	MAJOR	fin-web01
▸ Transaction Time Alert	Sep 20, 2022, 9:03:44 AM	CRITICAL	fin-app01
▸ Disk Usage Too High	Sep 20, 2022, 9:03:14 AM	MINOR	fin-db01
▸ Slow Queries Time Alert	Sep 20, 2022, 9:03:14 AM	CRITICAL	fin-db01
▸ High Memory Usage	Sep 20, 2022, 9:03:14 AM	MAJOR	fin-db01
▸ Response Time Alert	Sep 20, 2022, 9:03:13 AM	MAJOR	fin-web01
▸ Interface Oper Status Changed	Sep 20, 2022, 4:00:07 AM	MAJOR	CFX-Prod-DR-Switch04
▸ Interface Oper Status Changed	Sep 20, 2022, 3:59:12 AM	MAJOR	CFX-Prod-DR-Switch02
▸ Interface Oper Status Changed	Sep 20, 2022, 3:59:07 AM	MAJOR	CFX-Prod-DR-Switch04
▸ Interface Oper Status Changed	Sep 20, 2022, 3:58:12 AM	MAJOR	CFX-Prod-DR-Switch02
▸ Interface Oper Status Changed	Sep 20, 2022, 3:56:11 AM	MAJOR	CFX-Prod-DR-Switch02
▸ Interface Oper Status Changed	Sep 20, 2022, 3:56:07 AM	MAJOR	CFX-Prod-DR-Switch04
▸ Interface Oper Status Changed	Sep 20, 2022, 3:54:11 AM	MAJOR	CFX-Prod-DR-Switch02
▸ Interface Oper Status Changed	Sep 20, 2022, 3:53:07 AM	MAJOR	CFX-Prod-DR-Switch04
▸ Interface Oper Status Changed	Sep 20, 2022, 3:51:11 AM	MAJOR	CFX-Prod-DR-Switch02
▸ Interface Oper Status Changed	Sep 20, 2022, 3:51:06 AM	MAJOR	CFX-Prod-DR-Switch04

Fig: Example of BizOps or Experience Desk dashboard

03. L1 or Command Center Dashboard

<u>Responsibility:</u> To ensure that incidents coming into service desk are triaged, diagnosed and resolved in a timely manner (in accordance with SLA). Typically follows the recommended knowledge base articles, or perform recommended diagnostic actions, and if the incident needs more specialized skills, escalates the incident to L2 or Engineering team.



CFX20220916240951: Total transaction time 55.65 ms exceeded specified threshold 50 ms, 10.95.134.102, fin-app01

Recommended KnowledgeBase Article: Troubleshooting Slow Transactions

Recommended Action: Investigate database resource utilization

Alert Type	Raised At	Severity
Disk Usage Too High	Sep 16, 2022, 9:49:07 AM	MINOR
Response Time Alert	Sep 16, 2022, 9:49:06 AM	MAJOR
High Memory Usage	Sep 16, 2022, 9:49:06 AM	MAJOR
Slow Queries Time Alert	Sep 16, 2022, 9:49:06 AM	CRITICAL
Transaction Time Alert	Sep 16, 2022, 9:48:31 AM	CRITICAL

Fig: Example of L1 or Command Center Dashboard

- <u>Incident Queue/My Queue:</u> Incoming incidents, my incidents and filter or query incidents based on various attributes. Ability to ingest trouble tickets from ticketing system (call-in or portal opened tickets)
- <u>Incident Room:</u> A curated and dynamically built portal to show incident-specific data, insights and tools at one place - to help accelerate incident diagnosis and resolution
- <u>Recommendation:</u> Recommended knowledge base article and recommended next best action
- <u>Service Health:</u> Full-stack map overlayed with operational alerts/events
- <u>Triage Dashboard:</u> Shows metrics and logs at the time of incident occurrence - pulled from-monitoring tools and log systems
- <u>Collaboration:</u> Share incident, diagnostic results or comments with L2 and application or infrastructure teams

04. L2/L3 or Engineering Dashboard

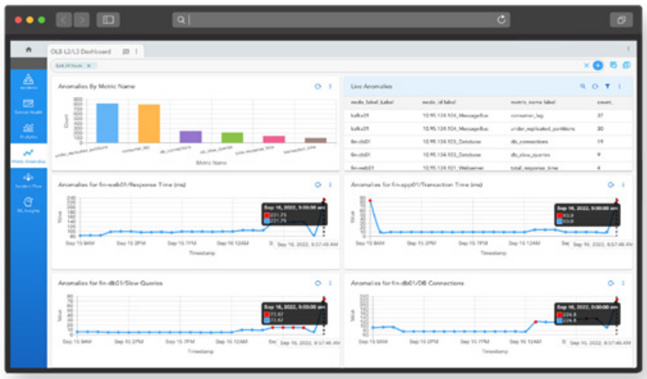


Fig: Example of L2/L3 or Engineering Dashboard

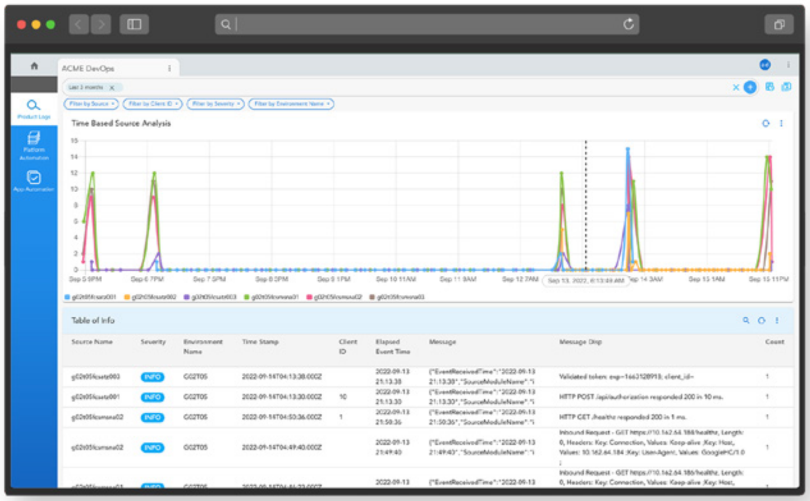
Responsibility: Incident root cause identification and problem resolution, by conducting thorough service health evaluation, operational data analysis, performing advanced diagnostics and/or remediation tasks, and collaborating with application or infrastructure teams. In addition to incident-specific data, this user role has access to more advanced data and insights that show historical trends about application and various components.

- Service Health: Full-stack service map overlayed with alerts/events. Ability to select a particular node in the map and view activity related to that node
- KPIs: Timeseries metrics for key operational performance data
- Metric Anomalies: Anomaly detection of KPIs, live anomalies and trend graphs
- Alert/Events Heatmap: 24-hour heatmap of alerts/events across the full-stack
- Incident Flow: Shows incoming alerts by source and correlated incidents by priority
- Incident Room: A curated and dynamically built portal to show incident-specific data, insights and tools at one place - to help accelerate incident diagnosis and resolution

DevOps Dashboard

Responsibility: Reduce triage time during DevOps issues (ex: release, CI/CD) and functional test failures. Ability to drill down to identify application artifacts (like logs, changes) that may have a causal impact on a test case failure or an incident that resulted because of a change.

- Platform Sanity: Summary of platform operations, test case failures and drill down by priority and type
- Application Logs: Application and microservice logs generated in response to CI/CD operations and test suite executions



Source Name	Severity	Environment Name	Time Stamp	Client ID	Elapsed Event Time	Message	Message Disp	Count
g02t05fcsetz003	INFO	G02T05	2022-09-14T04:13:38.000Z		2022-09-13 21:13:38	{"EventReceivedTime":"2022-09-13 21:13:38","SourceModuleName":"i	Validated token: exp=1663128918; client_id=	1
g02t05fcsetz001	INFO	G02T05	2022-09-14T04:13:30.000Z	10	2022-09-13 21:13:30	{"EventReceivedTime":"2022-09-13 21:13:30","SourceModuleName":"i	HTTP POST /api/authorization responded 200 in 10 ms.	1
g02t05fcsmsna02	INFO	G02T05	2022-09-14T04:50:36.000Z	1	2022-09-13 21:50:36	{"EventReceivedTime":"2022-09-13 21:50:36","SourceModuleName":"i	HTTP GET /healthz responded 200 in 1 ms.	1
g02t05fcsmsna02	INFO	G02T05	2022-09-14T04:49:40.000Z		2022-09-13 21:49:40	{"EventReceivedTime":"2022-09-13 21:49:40","SourceModuleName":"i	Inbound Request - GET https://10.162.64.188/healthz, Length: 0, Headers: Key: Connection, Values: Keep-alive ;Key: Host, Values: 10.162.64.184 ;Key: User-Agent, Values: GoogleHC/1.0 ;	1
[illegible]	INFO	G02T05	[illegible]		2022-09-13	{"EventReceivedTime":"2022-09-13	Inbound Request - GET https://10.162.64.186/healthz, Length: 0, Headers: Key: Connection, Values: Keep-alive ;Key: Host,	1

Fig: Example of DevOps Dashboard

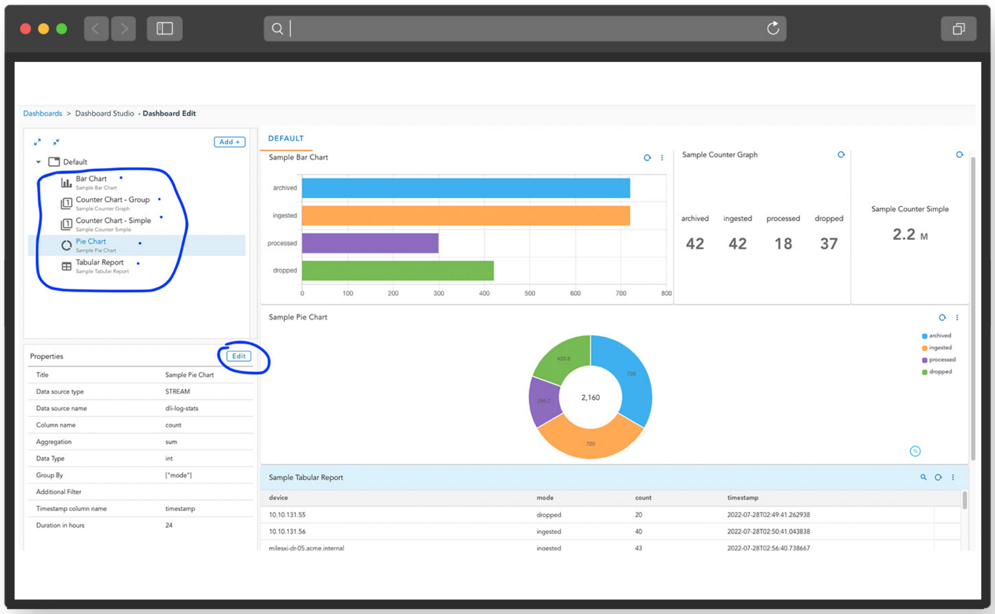
Composing Your Own Dashboards

You can easily compose your own dashboards based on the user persona, role and tenancy needs

- **Widgets:** Pick any out of the box widget from any application deployed in CFX platform, or create your own widget and define where it gets data from. Many different types of widgets are available in the system (and new ones are getting added frequently).
- **Data:** Widgets can typically pull data from persistent-streams or datasets.
- **Filters:** Dashboards can have data selection filters (like time, incident-state, priority etc.)
- **Layout:** Dashboard layout is self-adjusting based on widgets and the data in each widget, with additional capability to customize the layout bassed on min, max width and height settings.
- **Role:** Which user/role can access this dashboard? (user-id or user-role)
- **Scope:** What is the access scope of this dashboard? (project, workspace of business service)

01 Composing Dashboards with Dashboard Studio

In this utility, you can use UI-based designer to create a dashboard by selecting widgets, defining widgets, and defining sections/pages. You can use existing dashboards as a reference, and make iterative modifications until the desired dashboard is achieved.



Properties

Title	Sample Pie Chart
Data source type	STREAM
Data source name	dli-log-stats
Column name	count
Aggregation	sum
Data Type	int
Group By	["mode"]
Additional Filter	
Timestamp column name	timestamp
Duration in hours	24

Sample Tabular Report

device	mode	count	timestamp
10.10.131.55	dropped	20	2022-07-28T02:49:41.262938
10.10.131.56	ingested	40	2022-07-28T02:50:41.043838
milesxi-dr-05.acme.internal	ingested	43	2022-07-28T02:56:40.738667

Fig: Dashboard Composer Studio with Drag & Drop functionality

02 Composing Dashboards with JSON Definition

Some DevOps users and power users like to define or customize dashboards using JSON definition, which can be more convenient and preferred option for certain users. In this mode, the entire Dashboard is defined as ‘json’ blob. You can most certainly refer to existing dashboards and either take the full json definition or pickup sections, based on your needs.

The following two pictures show example of dashboard widgets that retrieve data from two primary data stores:

- **Persistent Streams** - for real-time streaming data with persistence (suitable for for events, logs, metrics etc.)
- **Datasets** - bounded datastores (suitable for inventory, configuration, dictionaries etc.)

Eg: Widget reads data from a Persistent Stream

```
“widgets”: [
   {
       “widget_type”: “custom_counter”,
       “title”: “Critical”,
       “stream”: “rda_synthetic_alerts”,
       “ts_column”: “timestamp”,
       “duration_hours”: 24,
       “min_width”: 2,
       “extra_filter”: “ severity is ‘Critical’”,
       “height”: 3,
       “max_width”: 4
   }
```

Eg: Widget reads data from a Dataset

```
“widgets”: [
   {
       “title”: “EC2 Instances by Type”,
       “widget_type”: “pie_chart”,
       “agg”: “value_count”,
       “column”: “InstanceId”,
       “dataset”: “cfx-aws-ec2-instances”,
       “extra_filter”: “`State.Name` is \”running\””,
       “formatter”: “CountFormatter”,
       “group_by”: “InstanceType”,
       “type”: “int”,
       “ts_column”: “LaunchTime”,
       “height”: 4,
       “min_width”: 3,
       “widget_id”: “7a589211”
   }
```

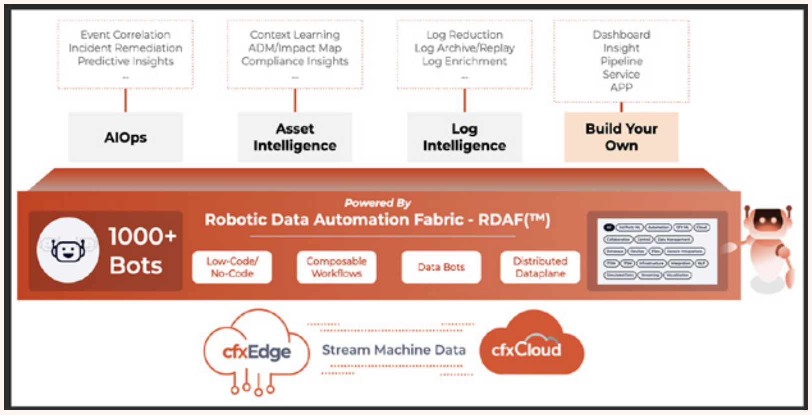
II. Composable Services

Most IT applications and workloads consume infrastructure, platform or software, as a service using IaaS, PaaS and SaaS providers. In the CloudFabrix platform, customers can consume data automations and data pipeline management as a service, to automate a set of complex IT functions and processes without worrying about underlying infrastructure and systems.

- Highlights: Easily build curated AIOps, MLOps, FinOps and DataOps services leveraging CloudFabrix Robotic Data Automation Fabric (RDAF) platform and cfxEdge capabilities.
- Benefits: Rapid time to value by having the ability to build desired services with the lowest cost and less specialized skills
- How it Works: Compose your own services in declarative language model (JSON format) using Service Blueprints, which define lifecycle management of pipeline andthe analytics to show

Out of the Box Premium Services

1. AIOps
2. Log Intelligence
3. Digital Asset Intelligence (FinOps)



Out of the Box Free Services

1. Demo Log Intelligence (Log Intelligence demo sandbox service with synthetic log data)
2. eBonding (ServiceNow to WebEx, PagerDuty, Twilio, OpenSearch and more)
3. AWS Dependency Mapper
4. RSS Feed to Slack
5. More ...

New services are being added to the platform frequently and check with CloudFabrix site for latest services.

Example of a Composable Service: eBonding

A Service in CloudFabrix AIOps platform is like any public cloud service or IT service, albeit comprises of the data, a set of pipelines that operate on data, and prescriptive analytics that show the results/stats of the service. A service has only backend data processing components without a frontend portal. Typically a service does data acquisition, collection, processing (filtering/cleaning/enrichment/transformation etc.), triggers actions, builds models, and finally sends out data to external tools.

For instance, eBonding is one such build your own service (BYOS) example. eBonding allows you to replicate incident data in real-time from ServiceNow to other ticketing/chatops tools like PagerDuty, WebEx Teams, Twilio, Jira ServiceDesk etc. You can also extend this service to perform bi-directional integration.

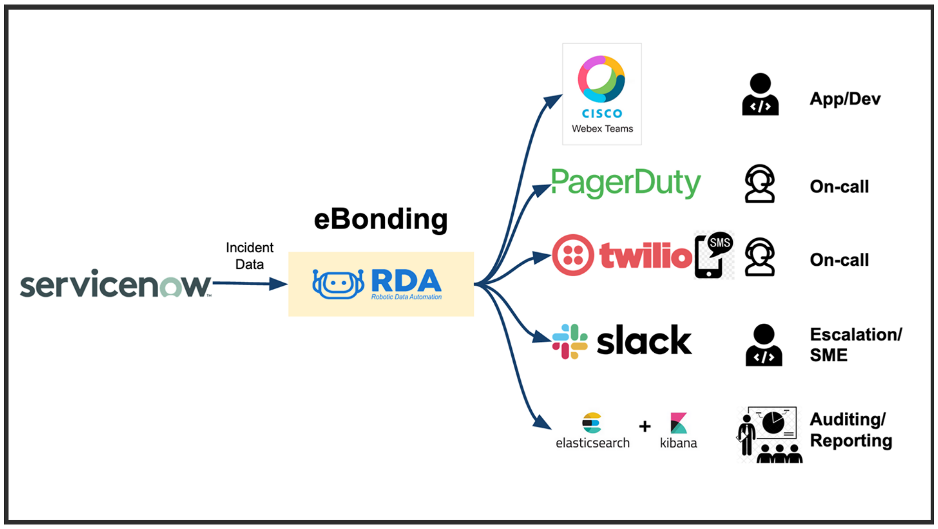


Fig: eBonding ServiceNow Incidents to WebEx, PagerDuty, Twilio, Slack and more.

Key Elements of a Service

Each service comprises of the following key elements:

- **Metadata:** name, id, description, version etc.
- **Pipelines:**
  - Action Pipelines: can be invoked on-demand to perform certain actions. For example: Run On-Demand Discovery Job, Trigger Data Collection, and Perform Diagnostic Task etc.
  - Service Pipelines: always running pipelines (akin to daemons in Linux parlance). Example, pipeline that listens to messages on a stream, pipeline that listens for change notifications from SQS etc.
  - Scheduled Pipelines: pipelines that can be scheduled to run periodically at certain intervals. For example: throttled discovery jobs, batch jobs etc.
- **Scaling Policy:** Elasticity of the service, min and max number of service instances
- **Worker/Site Affinity:** Where the pipelines should run
- **Analytics: Dashboard** that shows performance of service by visualizing data, jobs and service traces

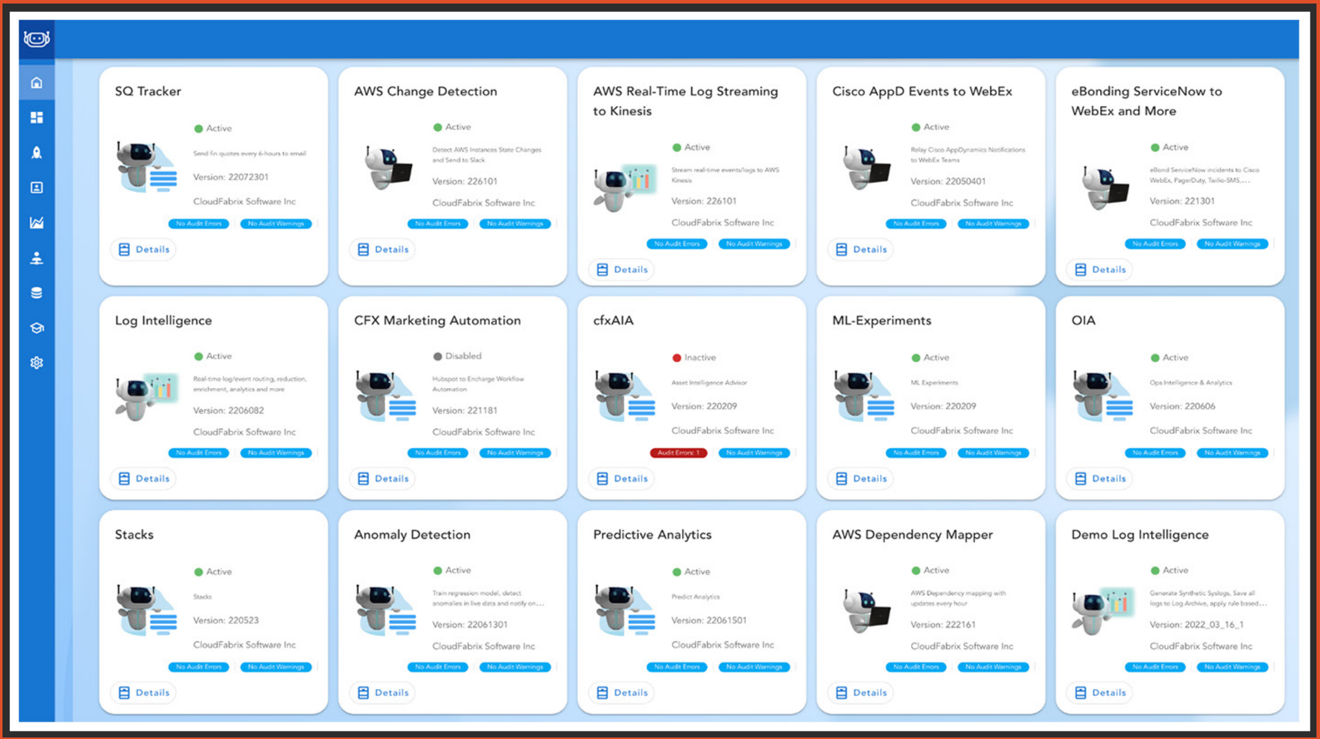
Composing your own Service with Service Blueprints

A Service is defined by a Service Blueprint, which is defined in JSON format with declarative model syntax. To begin with, you can copy the definition of any of the existing services and start making changes per your needs. Once you add a new Service Blueprint definition and enable it, then the Service starts running and performs its function as defined in the blueprint.

Example of a Service

Let us go through an example Service and see the key elements that we discussed.

- Service: AWS VM Change Detector
- Description: Listens for SQS notifications and send events to Slack
- Additional Details: Filters events for specific region (us-west-2) where Dev VMs are running



SQ Tracker
Active
Send fin quotes every 6-hours to email
Version: 22072301
CloudFabrix Software Inc
No Audit Errors | No Audit Warnings
Details

AWS Change Detection
Active
Detect AWS Instances State Changes and Send to Slack
Version: 226101
CloudFabrix Software Inc
No Audit Errors | No Audit Warnings
Details

AWS Real-Time Log Streaming to Kinesis
Active
Stream real-time events/logs to AWS Kinesis
Version: 226101
CloudFabrix Software Inc
No Audit Errors | No Audit Warnings
Details

Cisco AppD Events to WebEx
Active
Relay Cisco AppDynamics Notifications to WebEx Teams
Version: 22050401
CloudFabrix Software Inc
No Audit Errors | No Audit Warnings
Details

eBonding ServiceNow to WebEx and More
Active
eBond ServiceNow incidents to Cisco WebEx, PagerDuty, Twilio-SMS,...
Version: 221301
CloudFabrix Software Inc
No Audit Errors | No Audit Warnings
Details

Log Intelligence
Active
Real-time log/event routing, reduction, enrichment, analytics and more
Version: 2206082
CloudFabrix Software Inc
No Audit Errors | No Audit Warnings
Details

CFX Marketing Automation
Disabled
Hubspot to Encharge Workflow Automation
Version: 221181
CloudFabrix Software Inc
No Audit Errors | No Audit Warnings
Details

cfxAIA
Inactive
Asset Intelligence Advisor
Version: 220209
CloudFabrix Software Inc
Audit Errors: 1 | No Audit Warnings
Details

ML-Experiments
Active
ML Experiments
Version: 220209
CloudFabrix Software Inc
No Audit Errors | No Audit Warnings
Details

OIA
Active
Ops Intelligence & Analytics
Version: 220606
CloudFabrix Software Inc
No Audit Errors | No Audit Warnings
Details

Stacks
Active
Stacks
Version: 220523
CloudFabrix Software Inc
No Audit Errors | No Audit Warnings
Details

Anomaly Detection
Active
Train regression model, detect anomalies in live data and notify on...
Version: 22061301
CloudFabrix Software Inc
No Audit Errors | No Audit Warnings
Details

Predictive Analytics
Active
Predict Analytics
Version: 22061501
CloudFabrix Software Inc
No Audit Errors | No Audit Warnings
Details

AWS Dependency Mapper
Active
AWS Dependency mapping with updates every hour
Version: 222161
CloudFabrix Software Inc
No Audit Errors | No Audit Warnings
Details

Demo Log Intelligence
Active
Generate Synthetic Syslogs, Save all logs to Log Archive, apply rule based...
Version: 2022_03_16_1
CloudFabrix Software Inc
No Audit Errors | No Audit Warnings
Details

Fig: cfxCloud Portal Home Page showing multiple composed services

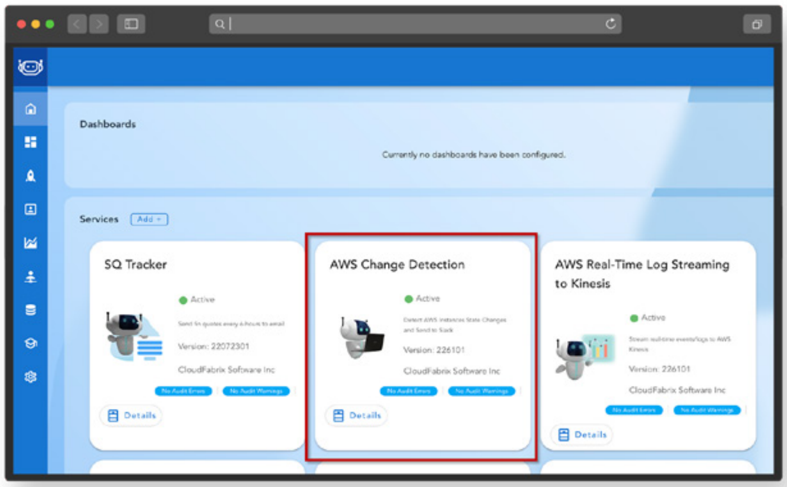
Service Catalog

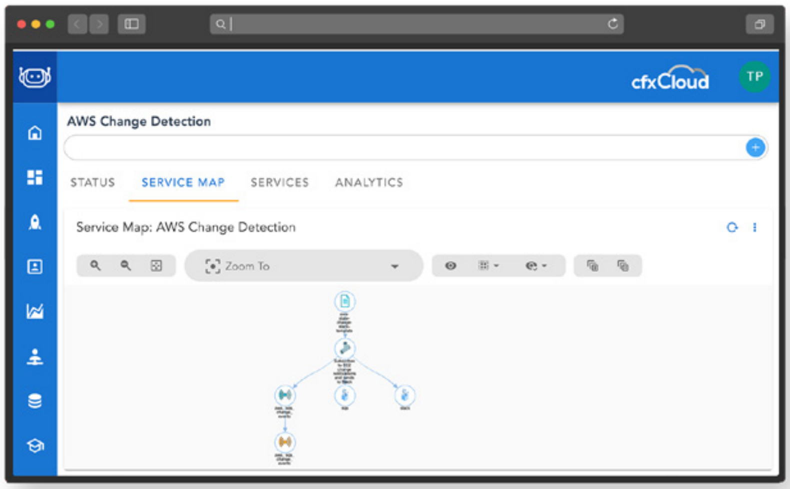
All services appear in CloudFabrix portal (cfxCloud) home page. Here you can see the catalog of installed Dashboards, Services & Apps.

Once you click on AWS Change Detection card it will launch the service dashboard that displays key elements of the service.

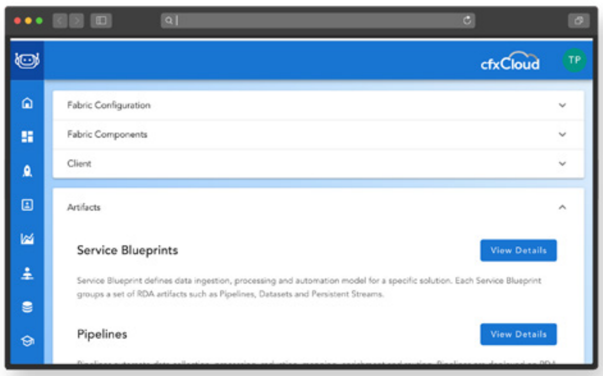
Once you click on AWS Change Detection card it will launch the service dashboard that displays key elements of the service.

- **Status:** Service status, audit errors, dependencies
- **Service Map:** Data lineage and tracing of various components that make up the service
- **Services:** Pipelines that make up the service
- **Analytics:** Dashboard that shows overall summary and statistics





Let us understand how this service is defined. You just have to add a new ‘Service Blueprint’ in the platform and it gets instantiated.



Fabric Configuration

Fabric Components

Client

Artifacts

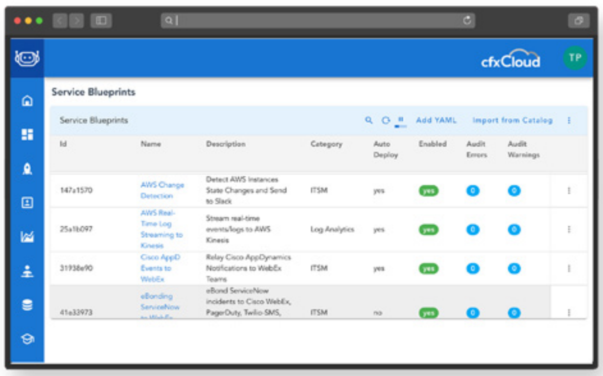
Service Blueprints

View Details

Service Blueprint defines data ingestion, processing and automation model for a specific solution. Each Service Blueprint groups a set of RDA artifacts such as Pipelines, Datasets and Persistent Streams.

Pipelines

View Details



Service Blueprints

Service Blueprints

Id	Name	Description	Category	Auto Deploy	Enabled	Audit Errors	Audit Warnings
147a1570	AWS Change Detection	Detect AWS Instances State Changes and Send to Slack	ITSM	yes	yes	0	0
25a1b097	AWS Real-Time Log Streaming to Kinesis	Stream real-time events/logs to AWS Kinesis	Log Analytics	yes	yes	0	0
31938e90	Cisco AppD Events to WebEx	Relay Cisco AppDynamics Notifications to WebEx Teams	ITSM	yes	yes	0	0
41e33973	eBonding ServiceNow	eBond ServiceNow incidents to Cisco WebEx, PagerDuty, Twilio-SMS,	ITSM	no	yes	0	0

Here is a snippet of Service Blueprint definition for this service, which is self-explanatory

```
“name”: “AWS Change Detection”,
“id”: “147a1570”,
“version”: 226101,
“category”: “ITSM”,
“comment”: “Detect AWS Instances State Changes
and Send to Slack”,
“enabled”: true,
“auto_deploy”: true,
“type”: “Service”,
“provider”: “CloudFabrix Software, Inc.”,
“attrs”: {},
“artifact-dependency-validation”: {
  “datasets”: {
    “verify”: true,
    “include”: “.*”,
    “exclude”: null
  }
},
“create_pstreams”: [
  “aws_sqs_change_events”
]
```

```
“service_pipelines”: [
   {
     “name”: “aws-change-detection-sqs”,
     “label”: “Subscribes to EC2 change
      notifications and sends to Slack”,
     “version”: “*”,
     “form”: {},
     “site”: “rda.*”,
     “site_type”: “regex”,
     “instances”: 1,
     “scaling_policy”: {
       “min_instances”: 1,
       “max_instances”: 4
     }
   }
 ],
 “action_pipelines”: [],
 “dashboard_sections”: [
   { …
   }
 ]
}
```

Ideas for composing your own Services:

You can potentially compose and build unlimited number of services on CloudFabrix platform, but just to help you give you a jumpstart, listing few ideas for services that you can use as a baseline and build something bigger and better.

- **CMDB Update:** listen for change notifications, trigger discovery and then updated CMDB
- **Log Archiver:** collect all IT logs in real-time, timestamp, compress and send to S3
- **Log Reducer:** collect all IT logs, filter, clean, truncate and send to SIEM/Datalake
- **Data Enricher:** enrich raw IT data by performing dictionary lookups or API queries
- **Text Analyzer:** run text through NLP and get sentiment, keywords or trouble tickets
- **AWS VM Change Detector:** listen for SQS notifications and send to Slack/MS Teams

III. Composable Pipelines

- **Highlights:** Easily acquire, ingest/integrate, prepare, enrich and transform data from any source, any tool and environment, like Edge, On-prem, Cloud and SaaS environments. Uses low-code/no-code approach of building workflows or pipelines that can use a wide variety of software data bots (1000+ bots in the library)

- **Benefits:** Provides for
    - Faster data integrations
    - Easy to implement changes
    - Field upgradability and agility
    - Operational savings (due to automation)

- **How it Works:** Bots are reusable code modules or functions that can be used over and over again. Bots perform specific tasks - like get VM metadata from AWS, post message to Slack channel, convert XML to JSOn and more. Pipelines allow you to sequence bots together, to automate more complex or real-world data tasks. Pipelines can be developed with low-code approach in a declarative model or with no-code using visual designer

Data Challenges in AIOps

AIOps platforms need to process vast amounts of data in order to learn from data, eliminate noise, identify the root cause and drive towards autonomous and predictive IT operations. Modern IT environments are expanding to datacenter, multi-cloud, edge, and hybrid environments – comprising a mix of modern and legacy tools that have many different data and API characteristics.

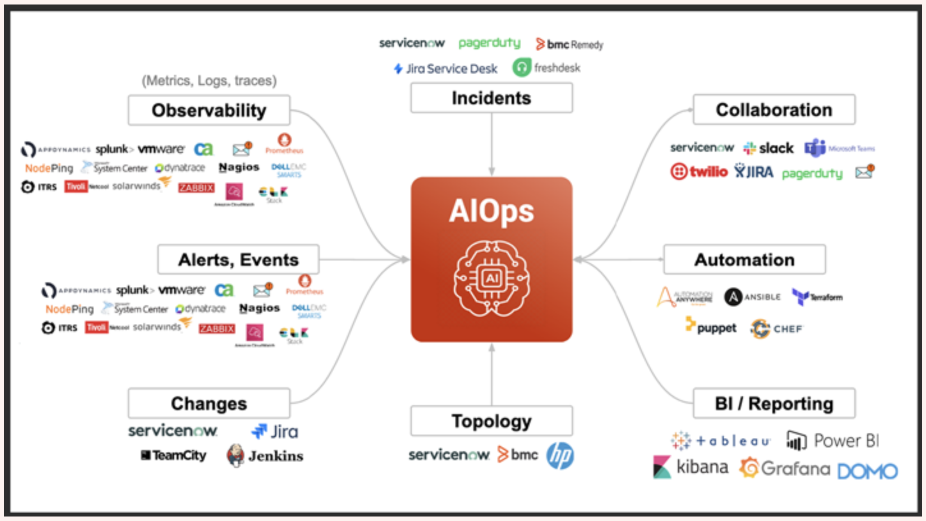


Fig: Complex Data Integration Ecosystem that AIOps needs support and participate in

Let us understand what are some of the key challenges in data preparation & data integration activities when implementing AIOps projects.

- **Varying data formats** (text/binary/json/XML/CSV), data delivery modes (streaming, batch, bulk, notifications), programmatic interfaces (APIs/Webhooks/Queries/CLIs)
- **Complex data preparation activities** involving integrity checks, cleaning, transforming, and shaping the data (aggregating/filtering/sorting)
- Raw data often **lacks application or service context**, requiring real-time data enrichment and bringing in context from external systems.
- Implementing data workflows requires a **specialized programming/data science skill set**
- **Changes** in source or destination systems require rewriting/updating connectors

Advantages of Composable Pipelines over Traditional Integrations

Low-code/No-Code data pipelines and bots-based integration has following advantages over traditional connector-based data integration.

- Ability to do dynamic enrichment at the time of ingestion - with lookups, API queries, data joins etc. Connector based allows only static key-value enrichments
- Event based data workflows (ex: on vmotion, trigger new VM metadata discovery) Data ingestion from edge environments, no need to open app-specific ports.
- Easily build custom data integrations for home-grown, legacy or adhoc systems.
- Ops personnel don't need to have programming skills or data science expertise to build pipelines and tailored data integrations – due to low-code/no-code
- Broad support for integration with cloud, databases, data warehouses, data lakes, message queues, streams, object stores, email, ITOM/ITSM tools, automation etc.

Effortless Query and Search on Object Stores, Datasets and Data Streams with CFXQL

CFXQL is a SQL like data query language, used to convey intent of what each bot should do with the data. All data bots support CFXQL. CFXQL makes it easy to interact with hundreds of different data sources and destinations in a unified manner. With CFXQL you can interact with Datasets, Databases, Persistent Streams and query or search over Object Storage, like MINio, AWS S3, Azure Blob storage, Google Cloud Storage and more.

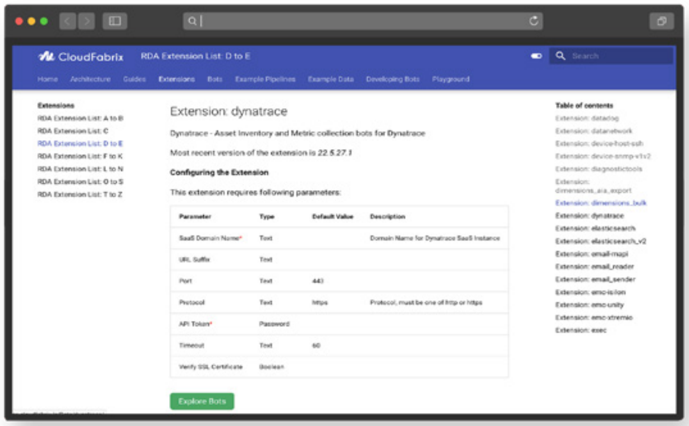
Following is an example of CFXQL statement that has SQL like semantics, which consists of two parts: Query and Result Format.

```
device_category is ‘SWITCH’ and
        ( severity is not in [ ‘INFO’, ‘DEBUG’
] or status != ‘CLOSED’ )
    GET
        device_category, severity, status as
‘Incident Status’
```

How Composable Pipelines Accelerate Data Integration

RDAF enables data integrations using low-code/no-code pipelines, featured extensions, and numerous data bots.

Key concepts/terminology: Think of a bot as a reusable code block or function that can perform an automated data task over and over again. For example, post-message-to-channel is a bot that posts input data frame to a specified Slack channel, and @kinesis:write-stream is another stream-processing bot that can write input stream to AWS Kinesis. One or more such related bots are packaged together as an Extension. For example, see Dynatrace extension & it has 11 bots



Extensions
RDA Extension List: A to B
RDA Extension List: C
RDA Extension List: D to E
RDA Extension List: F to K
RDA Extension List: L to N
RDA Extension List: O to S
RDA Extension List: T to Z

Extension: dynatrace

Dynatrace - Asset Inventory and Metric collection bots for Dynatrace

Most recent version of the extension is *22.5.27.1*

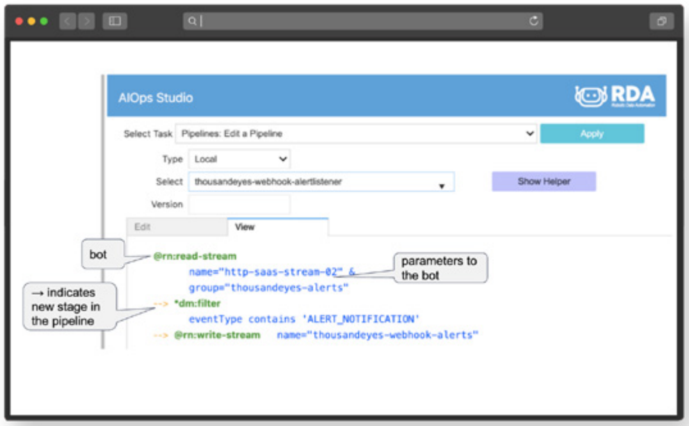
Configuring the Extension

This extension requires following parameters:

Parameter	Type	Default Value	Description
SaaS Domain Name*	Text		Domain Name for Dynatrace SaaS Instance
URL Suffix	Text		
Port	Text	443	
Protocol	Text	https	Protocol, must be one of http or https
API Token*	Password		
Timeout	Text	60	
Verify SSL Certificate	Boolean		

Table of contents
Extension: datadog
Extension: datanetwork
Extension: device-host-ssh
Extension: device-snmp-v1v2
Extension: diagnostictools
Extension: dimensions_aia_export
Extension: dimensions_bulk
Extension: dynatrace
Extension: elasticsearch
Extension: elasticsearch_v2
Extension: email-mapi
Extension: email_reader
Extension: email_sender
Extension: emc-isilon
Extension: emc-unity
Extension: emc-xtremio
Extension: exec

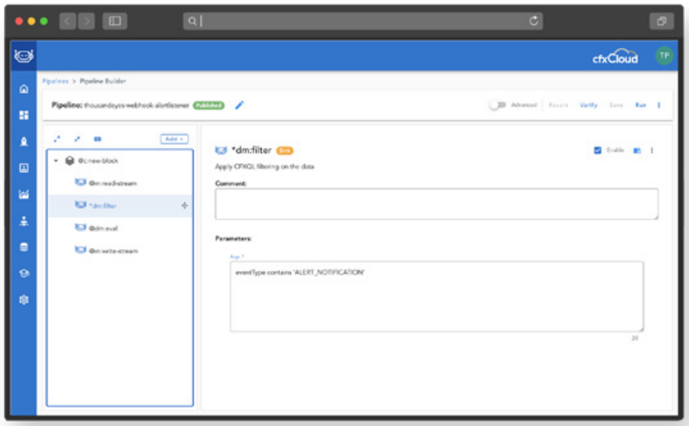
Bots can be sequenced together to execute and pass data from one bot to another, forming a directed acyclic graph (DAG) or a pipeline (aka workflow). The key here is that these workflows are implemented with a low-code/no-code declarative model, using YAML-like syntax. Comprehensive and real-world integration scenarios can be very easily developed by citizen developers, business technologists, and DevOps/ SRE personnel, without really having software programming or data science expertise.



Select Task: Pipelines: Edit a Pipeline
Type: Local
Select: thousandeyes-webhook-alertlistener
Version:

```
@rn:read-stream
        name="http-saas-stream-02" &
        group="thousandeyes-alerts"
--> *dm:filter
        eventType contains 'ALERT_NOTIFICATION'
--> @rn:write-stream    name="thousandeyes-webhook-alerts"
```

If you are wondering, what is this interface? This is our AIOps Studio, Jupyter Notebook-style visual workbench for authoring, testing, inspecting, and publishing pipelines. Every AIOps installation comes with this Studio. Think of pipelines as sequencing of bots and each bot take parameters, some mandatory, some optional Initially, you may find it difficult to pick the right bot for your task, but trust me, once you know which bot to use, you will start cranking out pipelines in no time. Curious to know more? explore some more example pipelines. We also have a no-code visual pipeline designer that allows for a drag-n-drop approach to building pipelines.



Pipeline: thousandeyes-webhook-alertlistener

@c:new-block
@rn:read-stream
*dm:filter
@dm:eval
@rn:write-stream

*dm:filter
Apply CFXQL filtering on the data
Comment:
Parameters:
Args *
eventType contains 'ALERT_NOTIFICATION'

Fig: Visual Drag & Drop Pipeline Builder

Bots and pipelines execute in data processing nodes called Workers, which are containerized software modules that can be installed anywhere, using docker-compose or Kubernetes helm charts. Workers form the core of the data plane, and every AIOps installation gets one or more Workers for data processing and pipeline execution. Workers can be deployed anywhere, for example, in remote branch sites, datacenters, aggregate sites, or your cloud VPCs, and essentially extend the fabric and allow you to ingest and integrate with local data sources without having to punch firewall holes in your corporate network.

All the worker nodes communicate over a secure, high-performance, and low-latency message bus that forms the core of the data fabric for AIOps. We use NATS.io as our data fabric and we made some customizations to suit our needs. NATS.io is being used by some renowned organizations like Tesla, Walmart, Paypal, Mastercard, VMware, AT&T, and more. NATS.io has some very impressive technology – check it out. I digressed, but back to RDAF – this data fabric, along with the scheduler, API gateway, and other key components form our control plane. You can separate the control plane and data plane, the way you like, and this is helping us implement many interesting hybrid scenarios.

The platform supports processing of the following types of Data via Data Integrations:

- **Datasets:** typically finite data, point in time. Examples: asset inventory, configurations etc.
- **Data Streams:** real-time, continuously generated data. Captured in a data lake. Ex: logs, alerts/events
- **Persistent Streams:** streaming data with persistence
- **Dependency Mappings:** Representation of connected assets using stacks that provide topology and dependency of assets or configuration items.
- **Log Archives:** UTC timestamped and highly compressed log files stored in any MinIO compatible object storage (ex: AWS S3, Azure blob storage, Google cloud storage, etc.)

Visit this link to learn more about these data types

Data Integration Scenarios:

Now that we have some basic understanding of RDAF and key terms let's discuss a few scenarios. Note that, in all these scenarios all the pipelines are completely field-customizable.

01. Featured Integrations

RDAF provides out-of-the-box integrations for many featured tools like AppDynamics, ServiceNow, Splunk, AWS CloudWatch, DataDog, Dynatrace, etc. We have out-of-the-box packaged pipelines that you can use to integrate and ingest observability data (metrics, events, logs and traces - MELT data). Check out our currently supported featured integrations

02. Custom Integrations

If your tool is not a listed integration for us, even then we can easily integrate with your tool using one of the following approaches, which in most cases your tool will support one or more of the following programmatic approaches of providing data to 3rd party tools.

A) Rest APIs:

If your tool supports HTTP/REST APIs, we have readily available bots that you can use for data integration.

- restclient bots
- httpclient bots

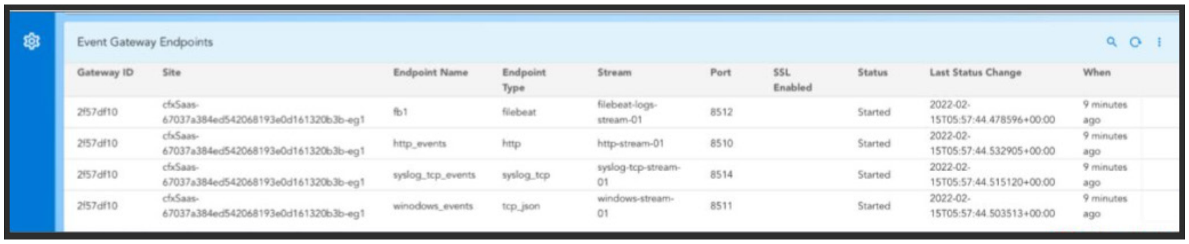
B) Graph QL queries:

If your tool supports graph-based queries, we also have readily available bots that you can use for data integration

- graphql bots

C) Webhook and More:

You can receive alerts/events/logs via Webhook using our stream processing bots. You can easily create new webhook endpoints and configure your monitoring tools to send alerts/ events to webhook. We also support remote webhooks i.e webhooks that are hosted in your corporate network – this can be achieved using our Event Gateway module, which is a special-purpose containerized module that can help you open many endpoints (like webhook, tcp, udp, syslog, rsyslog, tcpjson etc.) so that you don't have to open up your firewall ports to stream out data like logs/events.



Event Gateway Endpoints

Gateway ID	Site	Endpoint Name	Endpoint Type	Stream	Port	SSL Enabled	Status	Last Status Change	When
2f57df10	cfxSaas-67037a384ed542068193e0d161320b3b-eg1	fb1	filebeat	filebeat-logs-stream-01	8512		Started	2022-02-15T05:57:44.478596+00:00	9 minutes ago
2f57df10	cfxSaas-67037a384ed542068193e0d161320b3b-eg1	http_events	http	http-stream-01	8510		Started	2022-02-15T05:57:44.532905+00:00	9 minutes ago
2f57df10	cfxSaas-67037a384ed542068193e0d161320b3b-eg1	syslog_tcp_events	syslog_tcp	syslog-tcp-stream-01	8514		Started	2022-02-15T05:57:44.515120+00:00	9 minutes ago
2f57df10	cfxSaas-67037a384ed542068193e0d161320b3b-eg1	winodows_events	tcp_json	windows-stream-01	8511		Started	2022-02-15T05:57:44.503513+00:00	9 minutes ago

Fig: Webhook and other endpoints for real-time data ingestion

D) Message Bus integration – Kafka, NATS, AWS Kinesis, MQTT etc.:

It is not uncommon to see message-based data ingestion in very large environments, particularly in the financial sector, where high-speed telemetry data is delivered via message bus technologies like Kafka. Following bots and extensions can be used for data integration with messaging systems

- Kafka bots
- NATS streaming bots
- MQTT bots
- Stream processing bots

03. Custom Integrations – New Bot Development

If your monitoring tool or in-house solution doesn't fall into any of the above scenarios, and if you would like to develop a specific extension and bots for your tool, then this would be the scenario. In this case, we provide Bot Development SDK in Python, using which you can develop bots quickly. You can also work with the CloudFabrix presales team or with one of our implementation partners to have the bots developed for you. We have seen new bots being developed anywhere from 3-days to a week. Here are the resources for new bot development:

- Bot Development SDK

Using the RDAF platform, customers can cut down complex data integration tasks from months to days, and from hours to minutes, without really undergoing project delays and budget overruns.

Conclusion

A fundamental premise of AIOps is to be able to process vast amounts of data from X-domain sources and distributed environments, address multiple use cases and serve multiple stakeholder teams and user personas. The only way this can be achieved by an good AIOps solution is to have composability as one of its core principles.

- Composable Dashboards and Analytics: Serve multiple stakeholders (ex: executives, operators, administrators, experience desk, service desk, engineering, etc.)
- Composable Services: Address multiple use cases and provide curated services for multiple teams (ex: ITOps, ITSM, NOC, ServiceDesk, DevOps, CloudOps, etc.)
- Composable Pipelines: Ingest and process vast amounts of data from hybrid environments (ex: edge, branch, data center, multi-cloud, etc.). Integrate with featured tools, home-grown systems, legacy and adhoc systems.