StackState

Technical overview

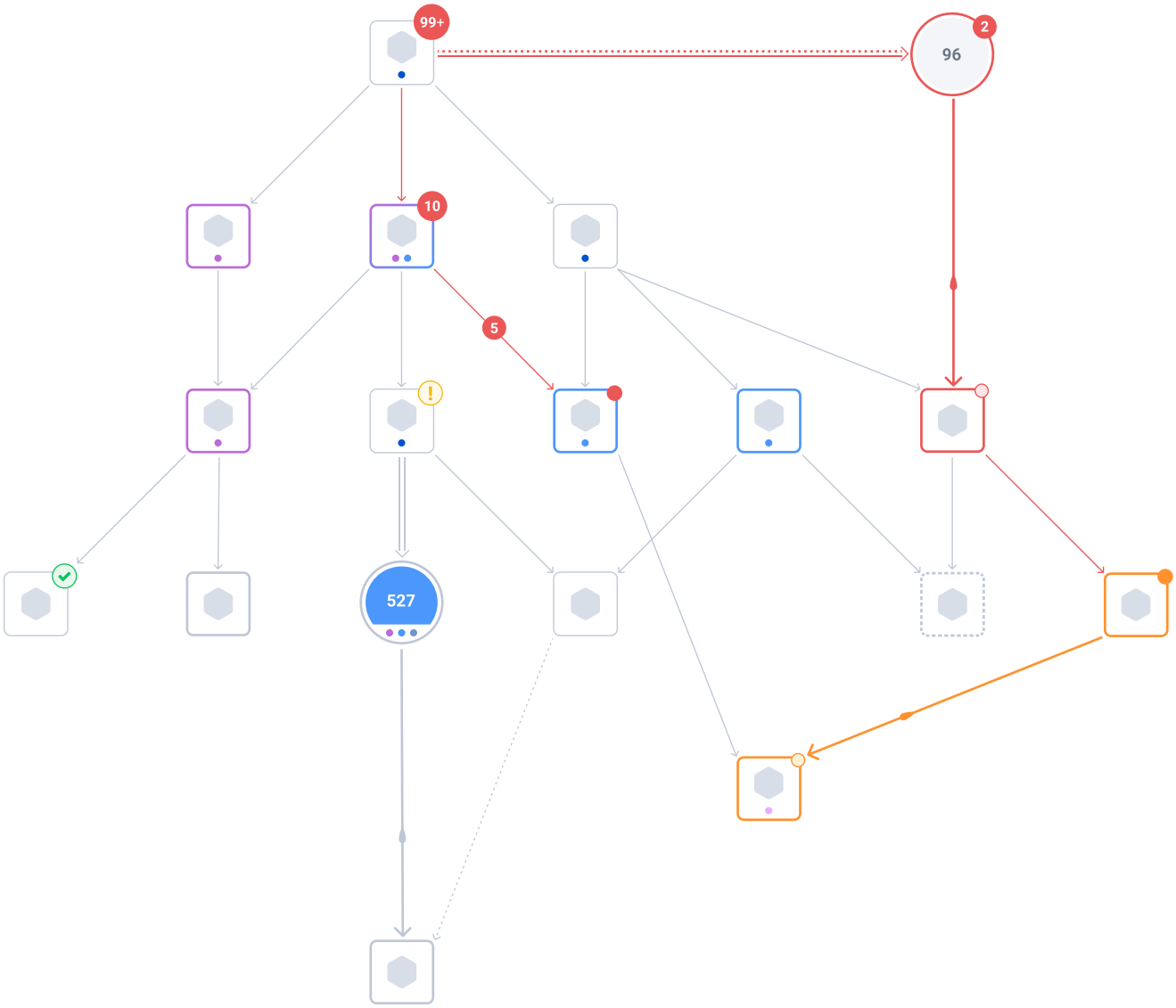
January 2021

Contents

About StackState

StackState delivers Relationship-Based Observability. The solution integrates with APM tools, infrastructure monitoring tools, virtualization and cloud platforms, Kubernetes, and incident management systems to add the certainty and richness of relationships, configuration changes, and AI-based diagnostics to existing incident management process. This uniquely leads to Deterministic Root Cause, which helps IT teams prevent and solve problems more quickly and efficiently.

This document includes answers to the most commonly asked questions about our technology.
Further detail can be found in our documentation https://docs.stackstate.com



Relationship-Based Topology. Note: topology shown differs from in-product topology.

4T Data Model

StackState's capabilities are driven by its unique 4T Data Model. By merging Topology, Telemetry, Tracing, and Time, we model the configuration of what is going on in any IT system. The 4T Data Model delivers insight into the entire IT landscape by capturing every millisecond of change, from any source in real-time to solve or prevent IT issues faster, with the right team.

The 4Ts stand for:

- **Topology**
- **Telemetry**
- **Traces**
- **Time**

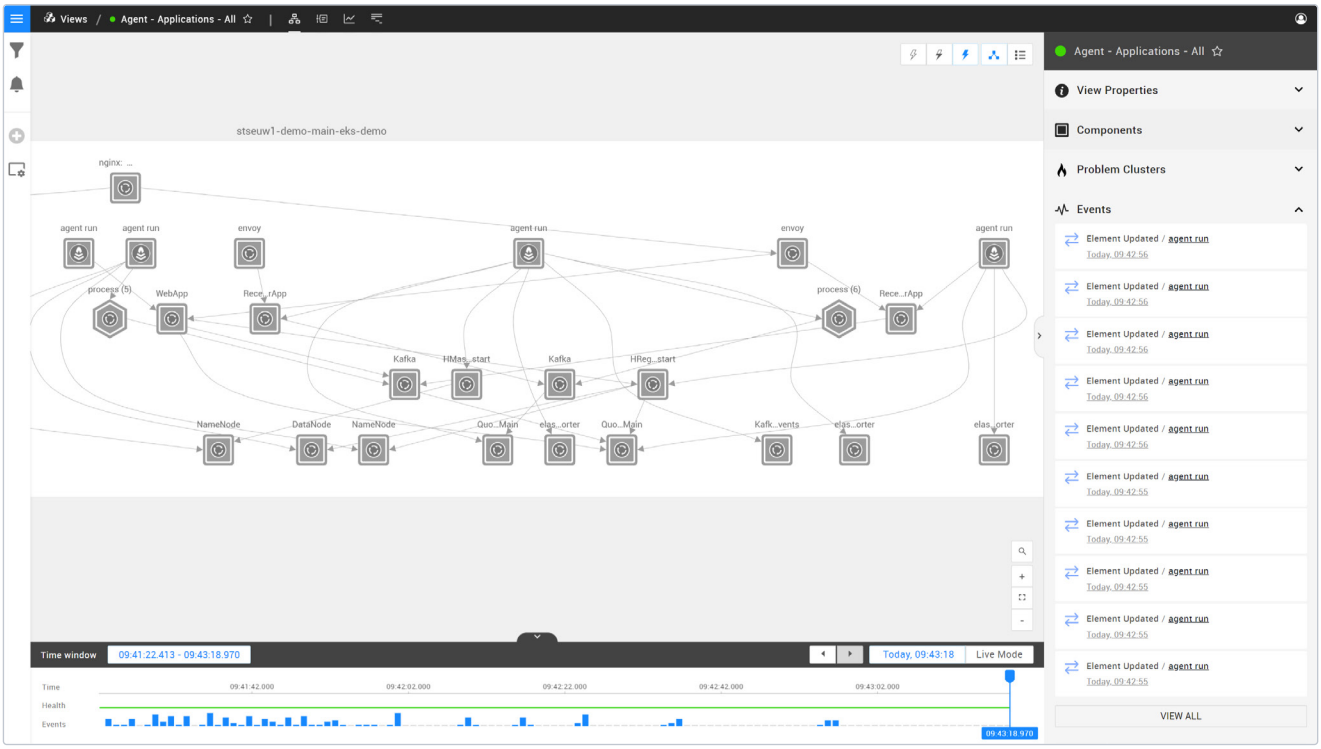
Topology

StackState combines data from a variety of data sources, deduplicating and merging it into one unified view. The platform integrates with the following types of sources to create the topology visualization:

1. **Manual data sources (e.g., CMDB, Excel-sheets, Asset Management Tools)**
2. **Discovery tools (e.g., ServiceNow Discovery, BMC or IBM Tivoli ADDM)**
3. **Tracing tools (e.g., AppDynamics, Traefik)**
4. **Automation/Deployment tools (e.g., Puppet, Chef, Ansible, Helm, MS DevOps)**
5. **Service Registries (e.g., AWS, MS Azure, Kubernetes, VMWare vSphere)**

With StackState's unique versioned graph database, the platform is able to "record" every change in your environment (up to the millisecond) and provide the ability to travel back in time, to see exactly what happened, where, when and why.

The data model of StackState is technology- and data-agnostic. This means a component in the topology could represent anything, from business application to hardware rack, from on-premise to cloud and from microservices to legacy systems.



Example topology visualization.

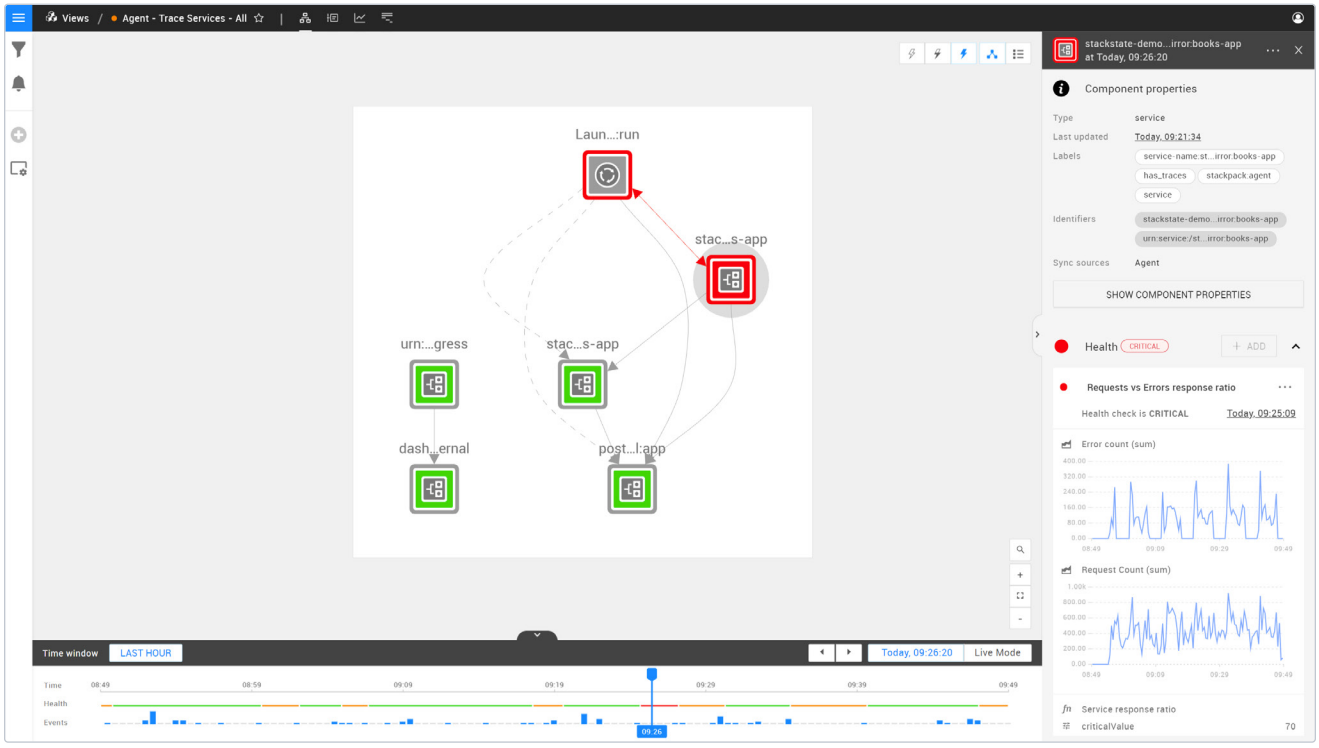
Telemetry

Telemetry typically comes from a variety of monitoring systems and can be divided into three types:

- **Metrics**
- **Logs**
- **Events**

Telemetry data is automatically mapped and consolidated to the components in the topology visualization. StackState applies a hybrid storage model: Telemetry data is polled from the original source, when needed. Also, telemetry data can be pushed to the StackState's API. This way the storage footprint of StackState is limited because there is no need to copy over complete data lakes or overloading APIs; this makes StackState unique from other observability tools that require you to port your data to their environment.

By combining topology and telemetry in StackState's 4T Data Model, you are able to understand the context between individual tools. This allows you to select the best of breed for all individual monitoring requirements, instead of relying on one solution from one vendor. StackState re-uses the information from these tools and automatically maps all telemetry (metrics, events, and logs) from these tools to the components and relations visualized in the topology visualization. This way the state of the components is represented.

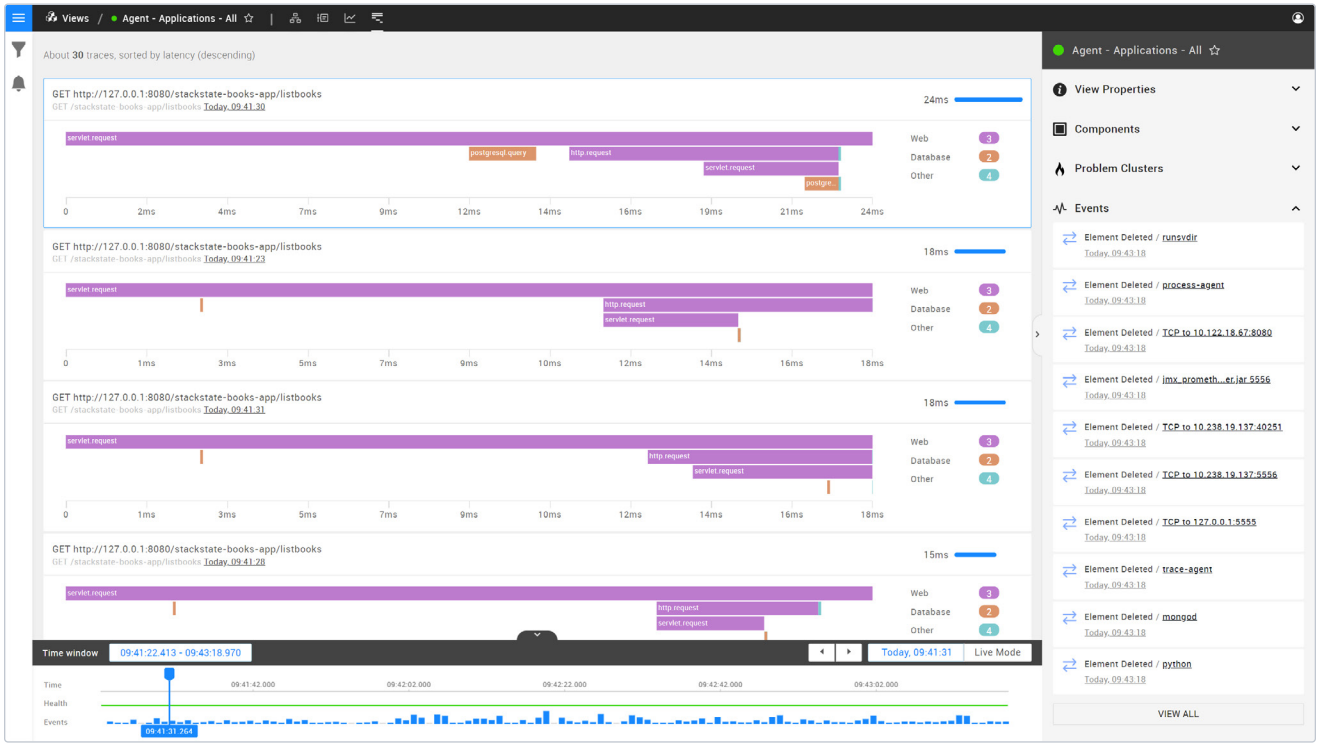


Example of telemetry automatically mapped to components.

Tracing

The StackState agent delivers tracing capabilities that gives you end-to-end insight into your entire IT landscape at the code level, with an easy-to-understand-and-navigate overview centered around the topology. Tracing is fully integrated with our unique time travel capability, which captures all changes over time.

StackState Tracing supports all major languages and has full support for distributed traces. StackState Tracing even integrates cloud tracing technologies such as Amazon X-Ray and Azure Monitor.

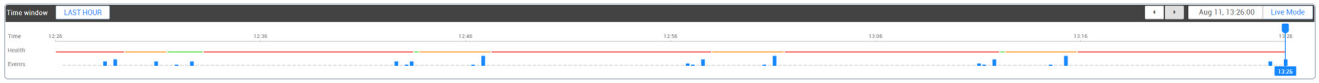


Example tracing perspective.

Time

Problems in IT stacks can usually be traced back to changes. Having a changelog of everything in your IT landscape is vital. To record these changes, we've built StackGraph, a versioned graph database. StackGraph allows you to go back to any moment in time and to see exactly what your landscape looked like at that moment.

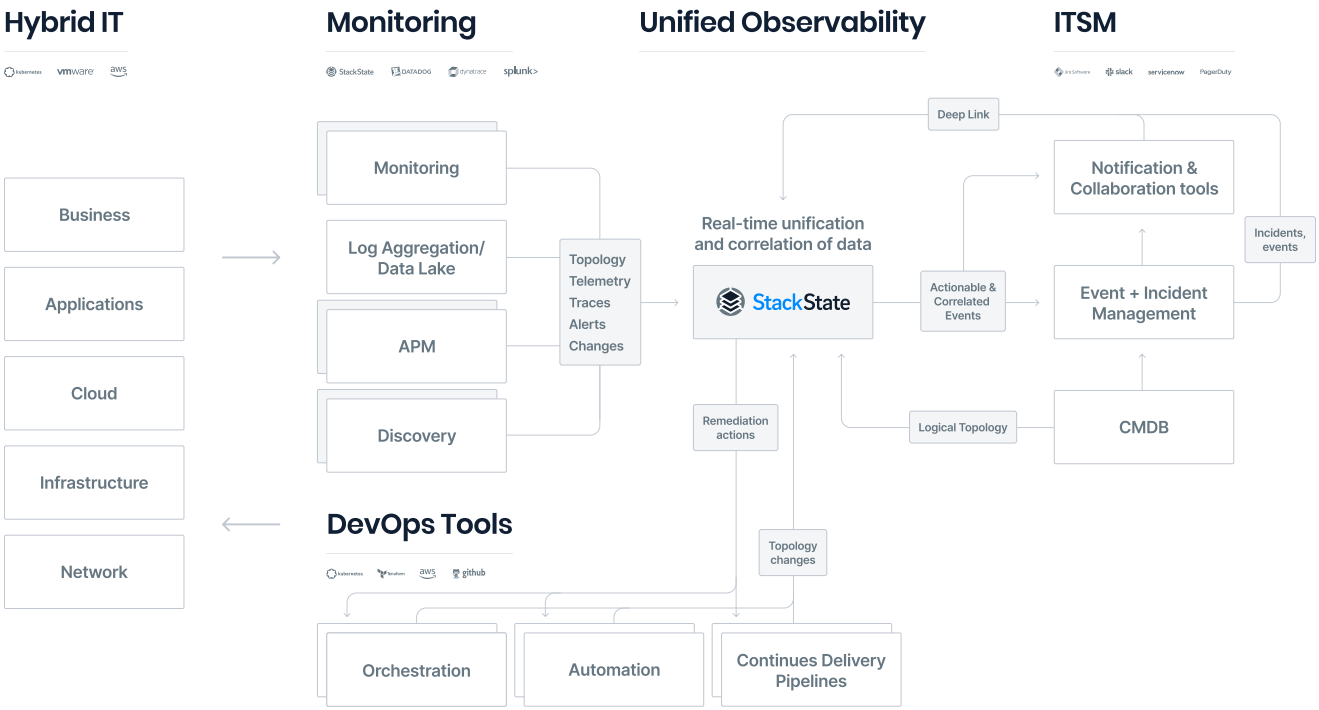
Traveling back in time is not limited only to topology, however. With streaming technology, StackState can also query the stores of the telemetry providers to request the data that belongs to the component and the moment in time selected. If the original telemetry store doesn't keep historical data, StackState can temporarily store that data for you.



Example of the timeline.

Where StackState Fits in Your Landscape

StackState respects the investments you have made in monitoring, log aggregation, and APM. By correlating topology, telemetry, traces, and all changes, StackState is able to reduce noise and provide context in to ITSM processes and automation workflows.



StackStates' Positioning Overview

Architecture

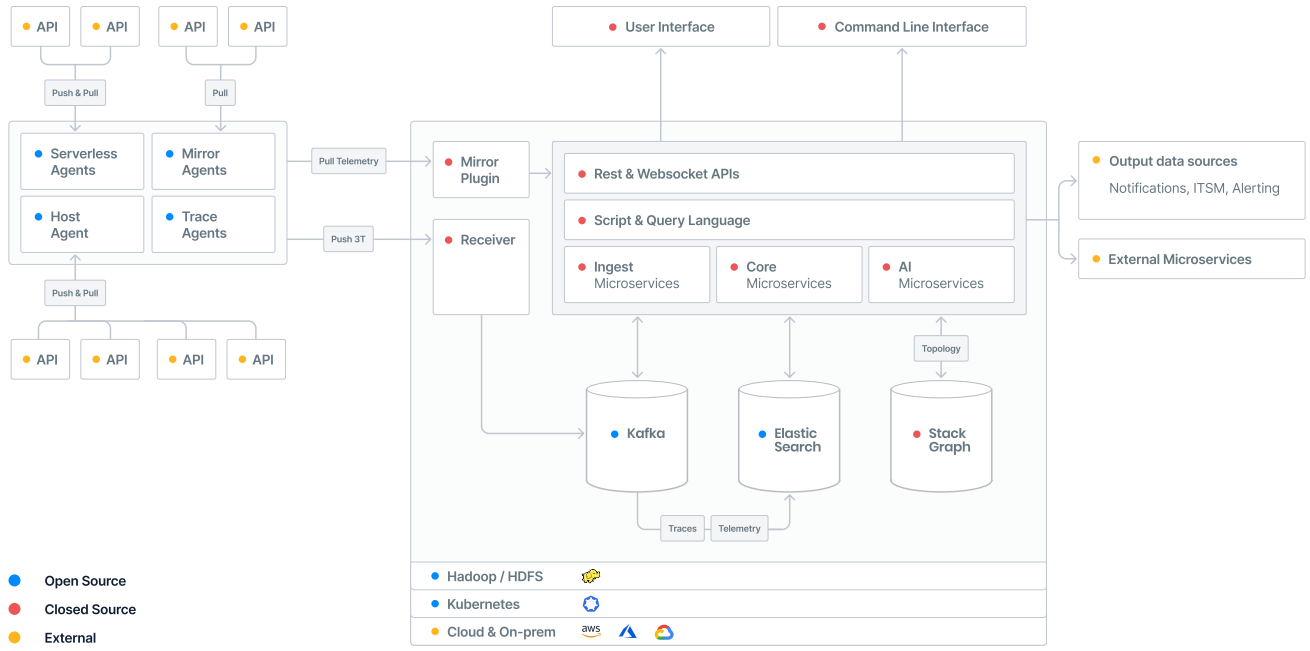
StackState is built for scale and runs on Kubernetes in your cloud or data center.

In most cases, a single Host Agent is installed on the StackState server to provide Agent-less integration with APIs from multiple sources. Data is gathered and received by one or more Agents and delivered at the Receiver API. From there, all data is put on Kafka. The data is processed by microservices and ends up as Topology in our versioned graph database, called StackGraph. Traces and some of the Telemetry data is temporarily stored in Elastic Search.

A Script and Query Language provides access to all dimensions of the 4T Data Model. They are also used by our own AI Microservices to interface with the 4T Data Model.

REST APIs are available for external services and are also used by our Command Line Interface.
Every user interface is kept up to date by Websocket APIs.

Alerts, notifications, tickets, webhooks, and API calls are just a few examples of output data sources to let you respond to situations you observe in StackState.



StackStates' Architecture Overview

Keeping the Topology Current

One of the most common questions we encounter is how the topology stays up to date. StackState has automated this process so there is no manual work required. Topology Synchronization allows you to automatically synchronize the topology of your stack to StackState based on information from such diverse systems as discovery tools, service registries, container management tools, and CMDBs.

A synchronization is defined by a data source and a number of mappings from the external system topology data into StackState topology elements. For a detailed overview, see **https://docs.stackstate.com/configure/topology/sync.**

The StackState Agent provides the following functionality:

- **Reporting hosts, processes and containers**
- **Reporting all network connections between processes/containers including network traffic telemetry**
- **Telemetry for hosts, processes, and containers**
- **100+ additional integrations**

More information on the StackState Agent can be found at **https://docs.stackstate.com/stackpacks/integrations/agent**

It is also possible to customize the StackState Agent and create your own Agent Checks. For more info, see **https://docs.stackstate.com/develop/developer-guides/agent_check/checks_in_agent_v2**

Integrations

StackState supports several integrations out of the box. Some of the most common integrations we see in our customer base include: AWS, Kubernetes, ServiceNow, VMWare vSphere, Azure, CloudWatch, Elastic, Prometheus, Splunk, Dynatrace, AppDynamics, and SolarWinds. For a full list and detailed overview, see **https://docs.stackstate.com/stackpacks/integrations.**

Deployment model

StackState can be deployed in you public/private cloud, on-premise, or as a SaaS offering.
Technical requirements for an on-premise deployment can be found here:
https://docs.stackstate.com/setup/requirements

Glossary

Here are the most common terms you'll encounter when learning about and using StackState.

- **4T data model** - stands for Topology, Telemetry, Traces, and Time. These four dimensions are the key concepts of the StackState data model.
- **Component** - the smallest unit of a topology; represents a real-life component in the IT landscape like a server, a service or an application. Each component belongs to one layer and one domain only.
- **Data source** - defines how StackState recognizes components of specified type during the topology creation process.
- **Event** - StackState records every change detected in the IT environment as an event.
- **Integration** - a link between an external data source and StackState as defined in a StackPack.
- **Layer** - represents a hierarchy that determines the relations and dependencies in your stack - typically top to bottom.
- **Relation** - models a dependency between components.
- **StackPack** - a package that is prepared for integration with an external data source.
- **Telemetry** - defined as either logs, metrics or events.
- **Topology** - consists of components and relations between those components.
- **View** - a partial visualization of the 4T data model that can be tailored to show only the cut of an IT landscape that is needed.