CNCF TAG App-Delivery Keptn Due Diligence

Promotion to CNCF Incubation

Authors:

- Jürgen Etzlstorfer, Dynatrace

Status of Document

- 2021-08-10: Initial version submitted to the CNCF TOC
- 2021-09-03: Adds TAG Recommendation, Jennifer Strejevitch
- 2021-09-07: Added issue for security TAG review, Jürgen Etzlstorfer

TAG Recommendation

Based on the due diligence the TAG can confirm that Keptn fulfils the formal criteria for incubation (this is subject to TOC user interviews to be performed). The project fills in gaps in several areas, including reducing the complexity of application and infrastructure delivery automation, strives for better integration patterns and works closely with the landscape projects for well defined integrations. It also applies and encourages the usage of cloud-native principles.

Key points to observe throughout the path to graduation:

- Ideally more contributions to be made by non Dynatrace affiliates
- The roadmap shows potentially big architectural changes in the future. It would be good to see how currently is the end user experience in relation to upgrades

Background

Link to TOC PR: https://github.com/cncf/toc/pull/670
Link to App Delivery TAG Issue: https://github.com/cncf/tag-app-delivery/issues/132
Link to Presentation: Google slides
Link to GitHub project: https://github.com/keptn/keptn

Link to Project website: https://keptn.sh/

Project Goal

Keptn is a cloud-native application life-cycle orchestration tool. Its goal is not to replace tools that are already present in an organization but to connect and orchestrate them to support the life-cycle management of applications. Keptn enables DevOps engineers as well as SREs to scale delivery and operations automation in their organizations.

Keptn builds upon declarative definitions for multi-stage environments, SLO-based quality gates, and auto-remediation and integrates with tools via an event-based approach.

Keptn CloudEvents: The event-based approach is built upon a well-defined set of *Keptn events*; currently in v0.2.2. All *Keptn events* conform to the CloudEvents specification in version v1.0. The CloudEvents specification is a vendor-neutral specification for defining the format of event data. In the course of the Keptn project, the event data is defined for the use-cases of application *delivery* and *remediation* as well as life-cycle orchestration tasks such as *deployment*, *test*, *evaluation*, *release*, *problem*, etc. The specification of Keptn CloudEvents is not limited to the mentioned tasks and can be easily extended by following the proposed format. The Keptn project is currently in the progress of aligning the *Keptn events* with the event specification from the Continuous Delivery Foundation (CDF) with the goal of establishing an industry-wide eventing standard for application life-cycle orchestration.

Keptn Control-Plane: Keptn is built for Kubernetes and consists of a couple of Keptn core services that altogether form the Keptn control-plane. The control-plane is responsible to orchestrate the life-cycle of an application managed by Keptn. Execution-plane service can connect to the control-plane to interact with Keptn via CloudEvents sent through NATS. The CloudEvents are currently stored in a MongoDB that serves as the datastore for all events that are sent via Keptn and allows for full traceability of life-cycle events. The architecture of the Keptn project can be found in the Keptn documentation.

Keptn Quality Gates: A central component of Keptn are quality gate evaluations based on service-level objectives (SLOs). Therefore, Keptn builds upon SRE best practices such as service-level indicators (SLIs) and allows to declaratively define SLOs for them. These SLOs define quality criteria for the applications and act as a gatekeeper during software delivery before promoting any application or microservice from one environment (e.g,. hardening) to the next environment (e.g., production).

Keptn Life-Cycle Orchestration: Keptn's architecture allows any tool to be integrated into the application life-cycle orchestration managed by Keptn. These *execution plane* services can run within the same cluster as the Keptn control plane or on different clusters, allowing to

orchestrate multi-cluster deployments, tests, evaluations, and operational tasks such as remediation orchestration or ChatOps.

Current Status

Progress since joining the CNCF Sandbox

Since joining the CNCF Sandbox, Keptn has made substantial progress in various dimensions, including user adoption, feature set, ecosystem growth, and community growth.

User Adoption: Keptn has been adopted by 25+ organizations (highlights being listed in section requirements) and seen a 6x increase of weekly downloads of the Keptn CLI. The successful usage of Keptn has been presented in Keptn User Groups, blogs, as well as presentations of Keptn users in meetups.

Feature Set: Significant improvements have been made to Keptn since the Sandbox stage. Some of them are highlights here:

- *Delivery assistant*: Promotion after an SLO-based quality gate evaluation can be done manually, automated, or even via 3rd party integrations using e.g., a ChatOps approach with Slack.
- *Support for closed-loop remediation*: Remediation sequences orchestrated by Keptn include evaluation of the executed remediation actions to verify if a remediation sequence can be aborted as a result of a fixed issue or might need escalation if automated remediation can not solve the issue.
- *Clear separation of control-plane and execution-plane*: Both components can be installed separately and allow for different execution-planes that are orchestrated by a Keptn control-plane. The separation of the planes enables:
    - *Parallel sequence executions*: The execution-planes are independent of each other meaning that they can execute sequences (e.g., for delivery or remediation) in parallel. This enables the use-case of deploying an app into multiple environments simultaneously.
- *Multi-cluster support*: Keptn control-plane and execution-plane can run on separate clusters in the same or even in different cloud providers, or on-prem. This allows, e.g., Keptn to orchestrate the delivery of the application even if each stage/environment is managed on a separate Kubernetes cluster.
- *Support for user-defined automation sequences*: Go beyond delivery and remediation by defining your own automation sequences including pre-defined and custom tasks.

- *Security improvements of Keptn*: Introduced RBAC for all Keptn core services to restrict permissions on the Kubernetes cluster, added security contexts for core services, and core services are not running root users anymore.
- *Change of tooling at runtime*: Developed an approach to manage the execution of Keptn integrations at runtime to allow for more flexible tool integrations. The subscription of tool integrations to the Keptn control-plane can be managed via a central interface.
- *Reduced default resource consumption & dependencies*: By removing dependencies in the Keptn default installation, the resource consumption could be reduced by ~55% in terms of CPU and memory resources. This could be achieved by a clear separation between control-plane (Keptn default installation) and the execution plane.

Ecosystem Growth: Keptn has grown its ecosystem by adding support for more than 10 tools and added and strengthened integrations with CNCF projects as well as other tools. Besides, the Keptn team is providing templates to foster new tools integrations.

- Added integrations since Sandbox: ArgoRollouts, LitmusChaos, Gitlab, Grafana, AnsibleTower, XMatters, Locust, Artillery, ZenDesk, Azure DevOps, OneChart (from Gimlet), Kubernetes Job-Executor, ArtifactHub, and counting (find all newly added integrations in the Keptn sandbox repo)
- Strengthened integrations with CNCF Ecosystem: CloudEvents, Prometheus, Helm, Jenkins

Community: We have significantly grown our community, on average by the factor >2 spanning across our multiple channels. In Slack, which is our preferred way to interact with the Keptn community, we have more than tripled our weekly active users (from around 50 to 180+), and all other community channels have shown a significant increase as shown in the following. Besides the community growth, we have established weekly Keptn developer and community meetings (see public CNCF calendar) as well as monthly Keptn user groups to foster exchange between Keptn adopters that share best practices on their Keptn usage.

Statistics	Sandbox	Current	Multiplier
GitHub Stars	410	1,026	**~2.5x**
Twitter followers	495	1,232	**~2.5x**
YouTube subscribers	78	272	**~3.5x**
Commits	3,751	5,994	**~1.5x**
Contributors	23	57	**~2.5x**
Slack weekly active users	50	180 (726+ in total)	**~3.5x**

Future Plans

The complete project roadmap is publicly available and upcoming building blocks are listed below:

- **Zero-downtime Upgrades & High Availability**: *Running Keptn at scale without downtime for end-users.*
    - Critical components in Keptn are running with multiple replicas, utilizing a rolling upgrade mechanism, and graceful shutdowns for various upgrading scenarios.
- **Keptn Uniform support**: *Seamless integration of DevOps tooling into the sequence orchestration by Keptn.*
    - UI-support for managing connected integrations and means to configure their subscriptions.
- **Security: Access Control**: *Allow fine-grained access control for interacting with Keptn.*
    - Basic functionality to control user access and user/API permissions on different entities (e.g., Keptn projects)
- **Alignment with Continous Delivery Foundation (CDF) eventing standard**: *Establish an industry-wide eventing standard jointly with the CDF.*
    - A translation layer will translate *CDF events* into *Keptn events*, followed by a step-wise integration of the established eventing standard into Keptn. Recording of SIG meeting, Link to PR, Link to Discussion
- **Remote management of execution plane**: *Central component (e.g., Operator) for installing and managing execution-planes.*

    - For handling multiple execution-planes that are connected to one Keptn control-plane, a central component (*in the sense of an Operator*) has to be in place that allows installing an execution-plane, handles the communication back to the control-plane, and is responsible for operating the integrations (which are running on the execution-plane).
- **Multi-tenancy**: *Reduction of resource footprint by multi-tenant capabilities.*
    - This concludes architectural changes in Keptn core for maintaining multiple clients with multiple projects by one Keptn deployment.

Project Scope

Clear project definition

Does the project have a clear and well defined scope

As already described in the project description section, Keptn's goal is not to replace tools that are already present in an organization but to connect and orchestrate them to support the life-cycle management of applications. Keptn enables DevOps engineers as well as SREs to scale delivery and operations automation in their organizations.

Therefore, the project targets three areas of automation:

- Observability, dashboards, and alerting by automating the configuration of these tools based on the Keptn shipyard (process definition) and SLO specifications
- SLO-driven multi-stage delivery by providing an abstraction layer for multi-stage delivery (shipyard) as well as built-in quality gates based on SLOs
- Operations and remediation, again based on a declarative process definition and built-in quality gates

Value-add to the CNCF ecosystem

Does the project have a clear value add to the current project ecosystem. How does it relate to other projects with overlapping capabilities.

Keptn brings clear value to the cloud-native ecosystem by orchestrating many of the tools already present in many organizations for deployment, testing, observability, and more.
By orchestrating these tools based on a process definition (shipyard), users of Keptn benefit from reduced complexity when managing existing applications and onboarding new applications to Kubernetes.

The Keptn ecosystem in terms of integrations is well aligned and integrated into the CNCF ecosystem, as will be outlined in the next section.

Alignment with other CNCF projects

In the following, we provide a list of ecosystem projects that have collaborated or integrated with Keptn.

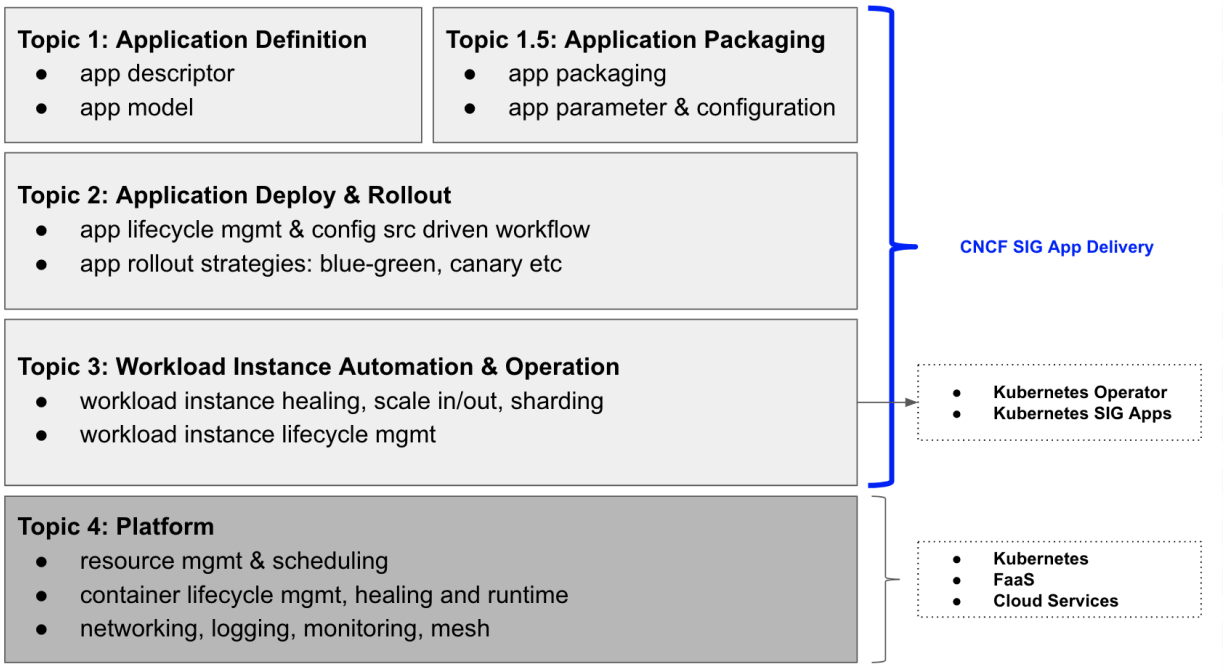
- Prometheus: configuring Prometheus by creating scrape jobs and alerting rules based on SLOs managed by Keptn and utilizing data from Prometheus for Keptn quality gates.
- LitmusChaos: orchestrating Chaos tests and load tests as part of a Keptn CD sequence as evaluating its impact with SLO-based quality-gates (presented at KubeCon 2021 Europe)
- Tekton: bridging the gap between CI and CD by defining interoperable events that can be exchanged between Tekton and Keptn (link to PR)
- ArgoRollouts: orchestrating a delivery sequence via ArgoRollouts for canary releases with Keptn quality gates to proceed or stop the rollout
- Crossplane: utilization of Crossplane for infrastructure management in DevOps workflows and Keptn CD sequences.
- Helm: deployment of applications via Helm charts orchestrated by Keptn
- Gitlab: automating deployment validation using Keptn's SLI/SLO-based quality gates orchestrated in Gitlab CI/CD pipelines.
- Istio: usage of Istio for traffic shifting between blue/green deployments. Keptn rewrites Istio virtual services and therefore manages the traffic shifting.
- CloudEvents: all events that are sent to and from the Keptn control-plane make use of the CloudEvents specification.
- OpenTelemetry: instrumenting Keptn with OpenTelemetry for full observability of events going through Keptn.
- NATS: Keptn is using NATS as its message system internally and contributed back to the NATS project.
- Grafana: Integration to automatically create dashboards for services managed by Keptn.
- Ansible: Integration to trigger Ansible Tower playbooks as part of Keptn's orchestrated remediation sequences.
- Jenkins: Jenkins shared library for integrating Keptn use cases with Jenkins pipelines.
- Auto-remediation working group: Keptn maintainers initiated a working group to discuss the current state and requirements of auto-remediation and to define the future of automated operations where auto-remediation is a big part of it.

Integrations that are currently planned for Keptn, in no particular order:

- ArtifactHub to provide a central hub for Keptn itself and Keptn integrations. Currently, integrations are curated via GitHub, and the ArtifactHub can provide a holistic view for the user on all available integrations. This is already work-in-progress.
- OpenTelemetry to align the Keptn's internal "Keptn context" with the trace-context supported by OpenTelemetry.
- ArgoCD for deployments as an alternative via Helm charts and ArgoRollouts that are already supported by Keptn.
- Flux, similar to ArgoCD, for deployments as an alternative via Helm charts and ArgoRollouts that are already supported by Keptn
- Falco to identify any security issues and have its ruleset managed by Keptn and use Keptn to orchestrate counter-action in response to identified security threats.
- Vault to support encrypted secret management in Keptn.
- Snyk to be integrated into the CD process to check for security vulnerabilities of container images as part of a quality gate and as a dedicated security scanning task during multi-stage delivery.

Alignment with App-Delivery Reference Model

Does the project align with the CNCF reference model and which capabilities does it require/provide at each level of the reference model.



As Keptn is an orchestrator of DevOps tools, it does not necessarily implement each of the 3 topics of the App-Delivery reference model but is closely aligned with and addresses them.
Topic 1 & 1.5: Keptn takes an Application Definition / Packaging description (such as a Helm chart) as input to orchestrate deployment.
Topic 2: Deployment models and rollout strategies such as blue-green or canary-releases can be orchestrated via Keptn. The definition thereof is manifested in the Keptn shipyard definition.
Topic 3: Regarding Workload instance automation Keptn has built-in capabilities to support the operation of applications in terms of providing an automation concept for any kind of remediation or self-healing actions, such as scaling, feature-flags, restarts, etc.

Formal Requirements

Document that the project fulfills the requirements as documented in the CNCF graduation criteria

Document that it is being used successfully in production by at least three independent end users which, in the TOC's judgement, are of adequate quality and scope

1. Schlumberger, an oilfield services company working in more than 120 countries, has currently 4 applications evaluated with Keptn quality gates. Evaluations make use of 10-20 SLIs, one of them even 90 SLIs per quality gate evaluation. Their integration triggers Keptn quality gates via Azure DevOps automation daily. It has been presented in a Keptn user group.
2. Vitality Group is a leading behavior change platform to make people healthier and enhance and protect their lives. They are triggering Keptn quality gates by utilizing an integration with Jenkins. Currently, Vitality has 22 services managed by Keptn in DEV, TEST & QA environments. They are running quality evaluations with Keptn multiple times a day in DEV, once per day in TEST, and ad-hoc via the Keptn API in QA.
    - Reference: can be shared privately with the TOC and will be shared here once it is public
3. Kitopi, the world's leading cloud kitchen platform, is using Keptn to evaluate the resilience of their applications. Using Keptn quality gates and Prometheus data, their evaluations are running in several nightly runs over 3 chaos stages, summing up in a total of over 700 runs as of March 2021. Find a reference of Adrian Gonciarz, Lead QA Engineer, who presented their use case.
4. Intuit, a leading provider of financial software in the United States of America, is using Keptn in conjunction with other tools in a system called DISTRO to allow for distributed load testing and evaluations thereof. The tooling includes Gatling, Argo, and Keptn.

Sumit Nagal, Principal Engineer, has summarized their application a blog and presented it in a Keptn user group

5. Raiffeisen Software is a leading software manufacturer for financial applications in Europe. They are running Keptn quality gates for internet banking, both for their mobile app and desktop applications, evaluating 30+ SLIs and SLOs for each release. The software quality is evaluated weekly via load tests, and integrated into Jenkins. Find their story here.

There are many more adopters of Keptn and even more organizations are currently evaluating the project for their production usage. Organizations that agreed to have them listed publicly can be found in the ADOPTERS.md file which currently lists 10 adopters, while others have only consented to share their details privately with the TOC, if required.

Have a healthy number of committers

Keptn currently has contributions from more than 15 different organizations and a total of 50+ individual contributors to the core project.

There are currently 3 levels of membership, with 12 maintainers at the top. A clear membership definition on how to become a member, approver, and maintainer of the project is available. Example of recent members that have been promoted via this process can be found here and here.

Due to the architecture of the project, it is worth mentioning that contributions and enhancement to Keptn do not necessarily be contributed to Keptn core, but often are contributed to the Keptn ecosystem in terms of Keptn-Contrib and Keptn-Sandbox. Both ecosystem repositories currently hold 30+ integrations in total, maintained by 65+ contributors (contrib / sandbox).

In addition to a clear membership definition, the Keptn project plans to establish a steering committee that helps to shape the future of the project and to provide guidance for maintainers.

Demonstrate a substantial ongoing flow of commits and merged contributions

The project averages at ~620 contributions from around ~16 contributors per month according to keptn.devstats.cncf.io contained within 126 merged PRs on average per month for the last year.

A clear versioning scheme

The projects versioning scheme is related to the semantic versioning structure and implemented for Keptn as follows:

- Major: currently 0 as the spec is actively developed and not yet finalized
- Minor: is incremented if breaking change in the Keptn spec occurs

- Patch: can contain bug fixes and features (including Keptn Enhancement Proposals)

The Keptn project has a 4 weeks release cadence since version Keptn 0.8.0. Before that releases were based on value increments. For our releases, Keptn maintainers follow the release checklist as documented for the latest releases 0.8.1, 0.8.2, and 0.8.3 where a dedicated release manager is responsible for driving the release process. If need be, there is a hotfix checklist that has been executed for a hotfix release in the past.

Other Considerations

Cloud Native

Does the project meet the definition of Cloud Native? The CNCF charter states:

> *"Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.*
>
> *"These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil."*

Yes, the Keptn project is well aligned with the definition of Cloud Native. It runs on Kubernetes and manages the lifecycle of cloud-native applications.
Aligned with Kubernetes, Keptn bases all of its configuration in a declarative (YAML) based manner.

The Keptn shipyard definition provides an abstraction of a delivery sequence, allowing the user to decouple the process definition from the tooling resulting in reduced complexity of delivery sequences and even allows to add, or change tooling at runtime without changing the delivery process.

Project and Code Quality

Are there any metrics around code quality? Are there good examples of code reviews? Are there enforced coding standards?

Keptn makes use of several tools to ensure high code quality, including Go Report, CodeCov, SonarCloud and others. In the following we presenting some of the statistics and will list recent PRs to demonstrate usage of these tools as well as how code reviews are done within the Keptn project.

Ratings:

- Go Report: A+ rating
- CII best practices: currently 99 % achieved

For each PR, the CI process (via Github actions) is triggered including checks with *CodeCov* to check code testing coverage and *SonarCloud* to identify any security vulnerabilities.

Some examples for pull requests in the Keptn project which highlight the integration with automated code quality checks and security scans as well as code reviews by the Keptn maintainers:

- PR 4840: https://github.com/keptn/keptn/pull/4840
- PR 3264: https://github.com/keptn/keptn/pull/3264
- PR 4737: https://github.com/keptn/keptn/pull/4737
- PR 1887: https://github.com/keptn/keptn/pull/1887

To help new contributors getting started with contributing to Keptn, the project is providing a Keptn developer documentation that gives recommendations for tooling, and details the code contribution and promotion workflow.

What are the performance goals and results? What performance tradeoffs have been made? What is the resource cost?

Keptn is not a computationally or storage intensive software and is intended to run alongside other pods and components on Kubernetes. Keptn is a control-plane for managing and orchestrating the life-cycle of applications running on Kubernetes and for this task, high throughput is not expected and therefore it is not regarded as a performance-critical component. However, to address potential performance issues, the messaging system of Keptn is based on NATS which is considered as high performant and production ready. To address resource consumption issues, each pod of the Keptn control plane is run with limited resources defined as resource requests and limits in the deployment specifications (can be found for example attached in the release notes or in the Keptn documentation).

Performance impacting issues that have been identified have already been addressed and fixed in the past (eg., issue 3969, issue 2905, issue 1901). Currently, Keptn maintainers are working on a high-availability setup for Keptn as well as zero-downtime upgrades when updating a Keptn installation (cf. KEP 48).

What is the CI/CD system? Are there code coverage metrics? What types of tests exist?

The project uses Github actions for its CI and deploys Keptn with Keptn. Each PR is automatically built and several tests, including unit-tests and checks by ReviewDog and SonarCloud. CodeCov metrics are generated for each PR as well. Integrations tests are run on a nightly basis, and if failing, a issue is generated automatically (example). More details on code metrics can be found in the previous section on code quality.

Has a security assessment by the security SIG been done? If not, what is the status/progress of the assessment?

A security assessment has not been done yet, but an issue has been raised.
Progress is being tracked here: https://github.com/cncf/tag-security/issues/784

Open Governance

How are committers chosen? How are architectural and roadmap decisions made? How many decision makers are outside the sponsoring organisation.

Keptn follows the governance document for new committers and has its community membership defined in a dedicated document. New Keptn contributors first enter the level of Members, before they get promoted to Approvers, and eventually Maintainers of the project. Currently, all maintainers (the highest level of our membership definition) are from one organization. However, our membership definition allows members/approvers to become maintainers by following the guidelines and satisfying the requirements. An example of a recent member that has been promoted via this process can be found here and here.

Vendor Independence

Is the project reasonably independent from the sponsoring vendor? Are all communication channels and project resources hosted just for this project or with other CNCF projects/resources? Is all code that is part of the project hosted and part of the CNCF managed orgs and repos? Are all defaults for upstream reporting either unset or community hosted infrastructure (i.e. doesn't point to vendor hosted SaaS control plane or analytics server for usage data)? Is all project naming independent of vendors?

All code (project source code as well as the website) is hosted in the keptn Github organization. The main communication channels for the project are Slack (via https://slack.keptn.sh as well as the #keptn channel in the CNCF slack workspace). Both Slack workspaces are open to join.

In addition to Slack, Keptn is hosting a website on keptn.sh (its source is available on Github) as well as a Twitter profile and LinkedIn page.
Keptn assets can be found in the CNCF artwork repository.

Relevant Assets regarding vendor independence

Keptn has been vendor-independent since its inception and, consequently, has been contributed to the CNCF. Keptn has already established an advisory board consisting of members from different organizations such as Red Hat, Gremlin, Civo, and end-user companies and is working on establishing a vendor-independent steering committee as well.