[SIG Observability] Due Diligence Project Review: Cortex Incubation

Authors: Goutham Veeramachaneni (Cortex Team)
Reviewed by: Bartek Plotka, Michael Hausenblas, Umair Ishaq (SIG Observability)
TOC sponsor: Katie Gamanji

Goals of this Document

This document provides a technical review of the Cortex project in a form of Due Diligence described here. The main goal is to provide factual data that would help TOC to decide if Cortex should be promoted to the Incubation stage.

Status

- 7-May-2020: *Document has been started and filled with initial data by the Cortex Team (Goutham Veeramachaneni) with collaboration of few SIG Observability members (Bartek Plotka).*
- 11-May-2020: *Document is shared on sig-observability mailing list before review meeting.*
- 12-May-2020: *As per task, the document is presented by the Cortex Team and reviewed by CNCF SIG Observability in the SIG meeting. Comments are added in the "SIG Observability comments" section.*
- *26-May-2020: Updates are presented to SIG ObsevabilityMay-2020: As per task, the document is presented by the Cortex Team and reviewed by CNCF SIG Observability in the SIG meeting. Comments are added in the "SIG Observability comments" section.*
- TBD: *If approved, the document is presented by SIG Observability at the TOC meeting.*

Statement of CNCF Alignment to TOC Principles

1. Project is self-governing

Yes, Cortex has a governance and team described here and here. Furthermore, the project has voting caps **(only 2 votes per company)** to make sure a single company doesn't take over the project.

SIG Observability Comments: SIG o11y agrees

2. Is there a documented Code of Conduct that adheres to the CNCF guidelines?

Yes, it's the same as the CNCF Code of Conduct.
https://github.com/cortexproject/cortex/blob/master/code-of-conduct.md

SIG Observability Comments: SIG o11y agrees

3. Does the project have production deployments that are high quality and high-velocity? (for incubation and graduated projects).

Yes, see https://github.com/cortexproject/cortex/blob/master/ADOPTERS.md. Further, at least two providers, Weaveworks and Grafana Labs, use Cortex as the backbone for their commercial offerings. Beyond that we have GoJek, EA, Rewe Digital using Cortex at massive scale (15Mil+ active series). We are working on producing case-studies for each end-user, with first case-study published here: https://cortexmetrics.io/docs/case-studies/

SIG Observability Comments: SIG o11y agrees (both velocity of adoption and of deploying from master)

Follow up questions: How Adopters are added? Submit a PR.

4. Is the project committed to achieving the CNCF principles and do they have a committed roadmap to address any areas of concern raised by the community?

What are the CNCF Principles? Is it this:

- Transparent, consistent technical and governance quality bar for graduation from incubation
    - See governance here: https://github.com/cortexproject/cortex/blob/master/GOVERNANCE.md and we have per company voting caps to make sure that a single company can't take over the project.
- Has users, preferably in production; is a high quality, high-velocity project (for incubation and graduated projects). Inception level projects are targeted at earlier-stage projects to cultivate a community/technology
    - See Question 3 which asks something quite similar
- Has a committed and excited team that appears to understand the challenges ahead and wishes to meet them
    - See a similar question and the one after that
- Has a fundamentally sound design without obvious critical compromises that will inhibit potential widespread adoption
    - See similar question
- Is useful for cloud native deployments & ideally, is architected in a cloud native style
    - Most of the deployments run on Kubernetes and are targeted at Kubernetes users.

- Has an affinity for how CNCF wants to operate
    - Can you elaborate on the question?

All the principles are of high merit and something we as the Cortex community aim for as well.

SIG Observability Comments: SIG o11y agrees

5. Document that the project has a fundamentally sound design without obvious critical compromises that will inhibit potential widespread adoption.

Cortex was built from the ground up to be highly scalable and so far has not hit any obvious scalability issues.

Potential areas of issues:

- Reliability
    - For static analysis, we have a strong test suite with a healthy mix of unit tests and end to end tests.
    - Weaveworks has CD enabled on the cortex repo and quickly catches any issues that might have crept in.
    - Grafana Labs does weekly internal releases from master. This means we should catch most bugs within a week as we're testing in production.
- Availability
    - All the components are highly available and you can configure the stateful components with the replication factor of your choice. See our answer on failure modes below..
- Security (Any sensitive data, any process for reporting those and handling?)
    - Cortex has experimental deletions support to remove any sensitive data that might have been stored.
    - The multi-tenancy model is documented here: https://cortexmetrics.io/docs/production/auth/
- Extensibility (Hardcoded backends? Plugins/Modules? Extensible APIs?)
    - We need two different kinds of backends for our storage: index (metric names, labels, which chunks to access) and chunks (actual time series data)
    - For index storage we support: Boltdb, Cassandra, Bigtable, DynamoDB.
    - For chunks storage we support: filesystem, Cassandra, Bigtable, DynamoDB, S3, GCS, Openstack Swift.
    - We're working on an experimental block storage based on Thanos which will remove the need for an index store. Going forward, the options for storage when using the block storage engine will be: filesystem, S3, GCS, Openstack Swift.
    - We're working on creating a plugin system for our storage engine on top of gRPC which will allow custom integrations with any backend.
- Scalability / Performance

    - We've worked on making Cortex highly scalable and highly performant over the recent times. See: https://grafana.com/blog/2019/12/02/kubecon-recap-configuring-cortex-for-maximum-performance-at-scale/ and https://grafana.com/blog/2019/09/19/how-to-get-blazin-fast-promql/
- Simplicity (Operational, Architectural, Automation)
    - Over the past year, we've worked specifically on making Cortex easier to use
    - We launched a website which lets them discover and share documentation much easier.
    - We made smaller-scale operations simple by allowing single-binary mode

SIG Observability Comments: SIG o11y agrees

6. Document that the project is useful for cloud native deployments & degree that it's architected in a cloud native style.

Cortex is used for several large-scale cloud native deployments. It's designed from the ground up in a cloud native style.

SIG Observability Comments: SIG o11y agrees, but requests tightening up the above section
2020-05-26: SIG o11y is happy with this answer

7. Document that the project has an affinity for how CNCF operates and understand the expectation of being a CNCF project.

We have team members who are also maintainers of incubating and graduated projects in the CNCF. From observing how the CNCF has evolved and how it works with older projects like Prometheus, we can confidently say that Cortex "has an affinity for how CNCF operates and understands the expectation of being a CNCF project". Some concrete examples:

1. We have a good governance model.
2. We have maintainers from different companies.
3. We have a regular public community call and all technical decisions are made in the public.

SIG Observability Comments: SIG o11y agrees, and requests expanding on the above section
2020-05-26: SIG o11y is happy with this answer

Review of graduation criteria and desired cloud native properties

Sandbox Graduation (Exit Requirements)

1. Document that it is being used successfully in production by at least three independent end users which with focus on adequate quality and scope defined.

- You can see the full list of end users as SIG o11y sees them here: https://github.com/cortexproject/cortex/blob/master/ADOPTERS.md
    - Aspen Mesh
    - Electronic Arts
    - GoJek
    - MayaData
    - Platform9
    - REWE Digital
    - SysEleven
- You can read about an end user-case study here: https://cortexmetrics.io/docs/case-studies/gojek/.
- WeaveCloud and Grafana Cloud offer Cortex as a paid hosted solution each operating at a massive scale
    - They have a lot of “end users” of the software as a service
    - But SIG o11y doesn’t consider these two companies as end-users in the context of this question

SIG Observability Comments: SIG o11y agrees

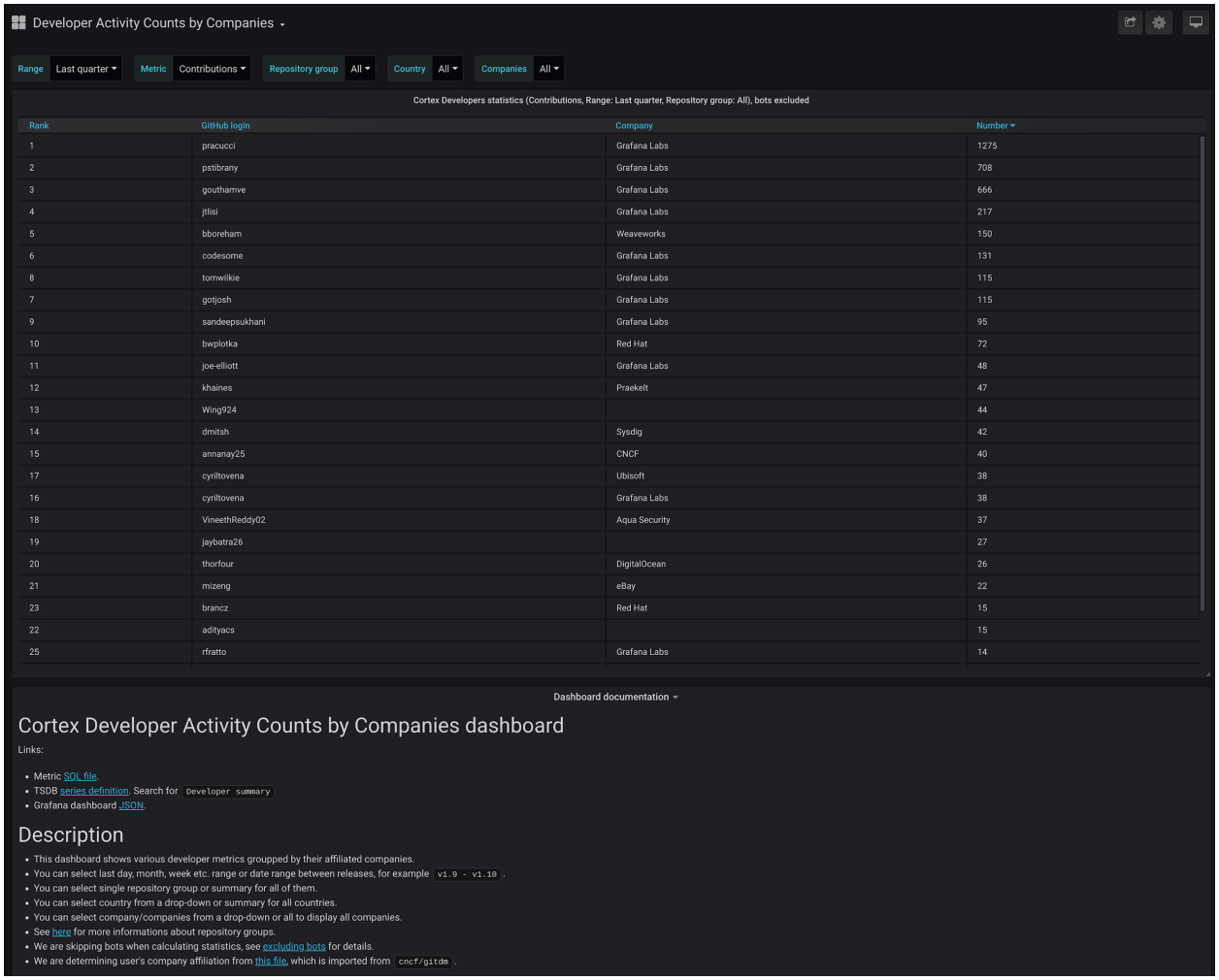
2. Have a healthy number of committers. A committer is defined as someone with the commit bit; i.e., someone who can accept contributions to some or all of the project.

We have 8 maintainers from 4 different companies. The details are at https://github.com/cortexproject/cortex/blob/master/MAINTAINERS.

Last Quarter Contributions:

Bryan Boreham, Weaveworks (@bboreham): **150**
Chris Marchbanks, Splunk(@csmarchbanks): **13**
Goutham Veeramachaneni, Grafana Labs (@gouthamve): **666**
Jacob Lisi, Grafana Labs (@jtlisi): **217**
Ken Haines, Microsoft (@khaines): **47**
Marco Pracucci, Grafana Labs (@pracucci): **1275**
Peter Štibraný, Grafana Labs(@pstibrany): **708**
Tom Wilkie, Grafana Labs (@tomwilkie): **115**

Source: link



SIG Observability Comments: SIG o11y agrees, Bartek will get more detailed numbers 2020-05-26: SIG o11y is happy with this answer

3. Demonstrate a substantial ongoing flow of commits and merged contributions. We are seeing a constant stream of performance improvements and features from the maintainers and community. See the stats here:

* Commits per week over the last 6 months
* Issue Opened/Closed per week over the last 6 months
* New PRs per week over the last 1 year

SIG Observability Comments: SIG o11y agrees

Documentation of CNCF Alignment (if not addressed above):

* name of project (must be unique within CNCF)

    Cortex

* project description (what it does, why it is valuable, origin and history)

    Cortex is a horizontally scalable, highly available, multi-tenant, Prometheus API compatible service that offers a long-term storage solution. Cortex started as a hosted Prometheus solution at Weaveworks to be part of Weave cloud before being donated to the CNCF. Freshtracks, Aspen Mesh and EA were some of the early adopters. Soon, Grafana Labs also started offering a hosted Prometheus solution on top of Cortex and started investing heavily in the project. The collective work of Weave, early adopters and Grafana Labs made Cortex scalable and fast and easy to use and increased adoption.

* statement on alignment with CNCF charter mission

    Cortex fully supports the CNCF's goal for scalability, "Ability to support all scales of deployment, from small developer centric environments to the scale of enterprises and service providers."

    There are many different ways to provide a scalable and available metric system for Kubernetes. Cortex with it's tenancy model combined with both the high-availability and horizontally scalability architecture serves this goal directly. Further, while having no dependency on Kubernetes, Cortex is built with Kubernetes in mind and most users deploy it in Kubernetes.

    We also provide a robust way for users to scale their Prometheus servers and Cortex has resulted in a lot of improvements in Prometheus itself.

* sponsor from TOC (sponsor helps mentor projects)

    Bryan Cantrill and Ken Owens (for sandbox application)

* license (charter dictates Apache 2 by default)

    Apache 2

* source control (GitHub by default)

    Github

* external dependencies (including licenses)

    Cortex depends on the following external software components:

    Prometheus (Apache Software License 2.0)

    Kubernetes (Apache Software License 2.0)

    Jaeger Tracing (Apache Software License 2.0)

    OpenTracing (Apache Software License 2.0)

    GRPC (Apache Software License 2.0)

Golang (Apache Software License 2.0)

* release methodology and mechanics
Done every 6 weeks following:
https://github.com/cortexproject/cortex/blob/master/RELEASE.md
* community size and any existing sponsorship
See: https://github.com/cortexproject/cortex/blob/master/ADOPTERS.md for adopters

SIG Observability Comments: SIG o11y agrees, and requests writing it down specifically
2020-05-26: SIG o11y is happy with this answer

Technical

- **An architectural, design and feature overview should be available.**

https://cortexmetrics.io/docs/architecture/

SIG Observability Comments: SIG o11y agrees

- **What are the primary target cloud-native use cases? Which of those:**
  - **Can be accomplished now:**

1. Horizontally scalable, long-term, global view of your metrics
2. Multitenant
3. Horizontally scalable alerting and recording rules
4. Backends leverage major cloud-native engines instead of baking their own

  - **Can be accomplished with reasonable additional effort (and are ideally already on the project roadmap):**

1. Removal of dependency on a NoSQL backend
2. Authentication system
3. Billing system

  - **Are in-scope but beyond the current roadmap for the next six months:**

1. Downsampling
2. Self contained system with no dependencies beyond compute (no dependency on even an ObjectStore).

  - **Are out of scope.**

SIG Observability Comments:

- **What are the current performance, scalability and resource consumption bounds of the software? Have these been explicitly tested? Are they appropriate given the intended usage (e.g. agent-per-node or agent-per-container need to be lightweight, etc)?**

We have tested (or are running in prod) clusters that supported more than 50Mil active series for extended periods of time. We have not seen any performance issues at that scale. See https://cortexmetrics.io/docs/guides/capacity-planning/ for resource consumption guidelines.

SIG Observability Comments: SIG o11y agrees

What exactly are the failure modes? Are they well understood? Have they been tested? Do they form part of continuous integration testing? Are they appropriate given the intended usage (e.g. cluster-wide shared services need to fail gracefully etc)?

Most of our components are stateless and can be scaled out independently. Let's break the failure modes into write and read path as they are fairly independent:

Write Path: We have distributors, consul, ingesters, backend db in the write path.

- Distributors are stateless services, and writes will succeed even if only one distributor is healthy.
- Ingesters store the data in-memory to compress them for 6-12h, and then flush the data to the backend store. You can read about the lifecycle and failure modes in detail here: https://cortexmetrics.io/docs/architecture/#ingester
- Consul stores the topography of ingesters and is critical. But consul is a HA service and in most configurations, can tolerate one or two of its nodes being down. But given writes and reads can tolerate consul being unavailable for 30s, we leverage this property by running it as a single replica and stateless, see: https://grafana.com/blog/2020/02/11/how-were-abusing-hashicorps-consul-at-grafana-labs/
- Backend Stores: We support DynamoDB, S3, Bigtable, GCS, Cassandra, Openstack Swift, and Minio as backend storage and they are either HA services or managed services.

Read Path: We have query-frontend, queriers, consul, ingesters and backend stores in the read path.

- Query-Frontend and Queriers are stateless services and queries will succeed as long as at least one replica of each is healthy.
- For the failure modes of the other components see above.

SIG Observability Comments: SIG o11y agrees

- **What trade-offs have been made regarding performance, scalability, complexity, reliability, security etc? Are these trade-offs explicit or implicit? Why? Are they appropriate given the intended usage? Are they user-tunable?**

We have focused on scalability and performance over simplicity in the beginning. This led to a complex architecture in the beginning which we've since strived to fix using a single-binary / single-process scale out approach.

One other trade-off we've made is to move fast in shipping features over documentation. This is something we're working to fix, we've made significant strides in recent times, but we still have a long way to go.

SIG Observability Comments: SIG o11y agrees that expectations have been met, but it needs to be written down
2020-05-26: SIG o11y is happy with this answer

[Bartek]: The only tradeoff is slight complexity, especially for the basic use cases. Cortex explicitly was done to satisfy large scale, multi-tenant use cases and that's when the price of operational complexity (lot's of components, tweakability) and resource consumption (e.g large amount of memory required for caching), truly pays off. This is why Cortex may stand out in terms of query performance even for large cardinality, with the price of complexity and leveraging on using NoSQL databases (although a cheaper option is being worked on!)

What are the most important holes? No HA? No flow control? Inadequate integration points?

- Biggest hole right now is that our stateful component: *ingester* cannot be scaled up and down super elastically, you have to add and remove them one at a time and addition / removal of an ingester takes a few mins.
- Good quality helm charts

SIG Observability Comments: SIG o11y agrees, not a blocker

Code quality. Does it look good, bad or mediocre to you (based on a spot review). How thorough are the code reviews? Substance over form. Are there explicit coding guidelines for the project?

[Bartek]: As part of Thanos maintenance I was both contributing and depending on Cortex code and I can definitely see huge improvement in code quality and quality of code review during the last half of the year. Before it used to be a fast moving project, but nowadays it's clear that the team has been working on the reliability very hard, focusing on decreasing sizes of their PRs and working on cutting edge e2e frameworks (which we use in Thanos now thanks to Cortex).

SIG Observability Comments: SIG o11y agrees

Dependencies. What external dependencies exist, do they seem justified?

You need a NoSQL store and ObjectStore for the data storage, and we're working on removing the dependency on a NoSQL store. You need a consistent KV store like etcd or consul for cluster coordination but we have added support for gossip to replace those. Finally you need redis or memcached for optional caching support.

SIG Observability Comments: SIG o11y is happy with the documentation provided

What is the release model? Versioning scheme? Evidence of stability or otherwise of past stable released versions?

See: https://cortexmetrics.io/docs/configuration/v1guarantees/ and https://github.com/cortexproject/cortex/blob/master/RELEASE.md

[Bartek]: Cortex just releases v1 and continuous on solid 6 week cadence. All releases are well documented and compatibility is well maintained with storage format versioning and migration code. Definitely seeing stability from my side.

SIG Observability Comments: SIG o11y is happy with the documentation provided

What is the CI/CD status? Do explicit code coverage metrics exist? If not, what is the subjective adequacy of automated testing? Do different levels of tests exist (e.g. unit, integration, interface, end-to-end), or is there only partial coverage in this regard? Why?

We have a strong CI running in CircleCI. We do extensive unit testing and integration testing. We don't have code coverage metrics, but we haven't seen major bugs in recent times.

[Bartek]: Definitely +100 on this. It would be nice to have code coverage numbers, but it looks like the majority of code is tested. Cortex is also maintaining a cutting edge e2e framework (which we use in Thanos now thanks to Cortex).

SIG Observability Comments: SIG o11y is happy with the documentation provided

What licensing restrictions apply? Again, CNCF staff will handle the full legal due diligence.

We use an Apache 2.0 license for all of our code.

SIG Observability Comments: SIG o11y is happy

What are the recommended operational models? Specifically, how is it operated in a cloud-native environment, such as on Kubernetes?

While we recommend Cortex to be run in Kubernetes, it doesn't have any Kubernetes specific dependencies at all and can be run on any environment.

SIG Observability Comments: SIG o11y is happy

Project

- **Do we believe this is a growing, thriving project with committed contributors?**

Yes, it is a very active project with many developers being paid to work on upstream.

- **Is it aligned with CNCF's values and mission?**

Yes, it was created to enhance cloud-native observability and is staying true to the mission.

- **Do we believe it could eventually meet the graduation criteria?**

YES!

- **Should it start at the sandbox level or incubation level?**

It is already in the sandbox, but should move to incubation.

- **Does the project have a sound, documented process for source control, issue tracking, release management etc.**

Yes, see: https://github.com/cortexproject/cortex/blob/master/RELEASE.md and https://cortexmetrics.io/docs/contributing/

- **Does it have a documented process for adding committers?**

Yes, see: Governance

- **Does it have a documented governance model of any kind?**

Yes, see: Governance

- **Does it have committers from multiple organizations?**

Yes, see: https://github.com/cortexproject/cortex/blob/master/MAINTAINERS

- **Does it have a code of conduct?**

Yes, see: https://github.com/cortexproject/cortex/blob/master/code-of-conduct.md

- **Does it have a license? Which one? Does it have a CLA or DCO? Are the licenses of its dependencies compatible with their usage and CNCF policies? CNCF staff will handle the full legal due diligence.**

Yes, Apache 2 and we do DCO

- **What is the general quality of informal communication around the project (slack, github issues, PR reviews, technical blog posts, etc)?**

We have an active slack channel #cortex on the CNCF slack where most of our users ask questions and discuss development. We are also quite active on GH and make sure our issues and PRs are responded to quickly.

- **How much time does the core team commit to the project?**

A significant amount of people on the core team are paid to work on Cortex and some people have Cortex as their full-time job. Infact, I can count collectively (adding partial involvement as well) 7+ people whose full-time job involves Cortex and its ecosystem.

- **How big is the team? Who funds them? Why? How much? For how long?**

We are 8 maintainers as detailed here: https://github.com/cortexproject/cortex/blob/master/MAINTAINERS and of these, people at Grafana Labs and Weaveworks are paid to work on Cortex as both companies run commercial offerings on top of Cortex. Other maintainers use Cortex at work or for their personal projects.

- **Who are the clear leaders? Are there any areas lacking clear leadership? Testing? Release? Documentation? These roles sometimes go unfilled.**

We have community members that are running Cortex in production and are constantly submitting features and improvements. This includes the active work on the next generation TSDB blocks based engine. Having said that, most of the improvements are in the code section and little in the documentation.

While we have a lot of things documented, we need a better structure for documentation and discovery. The active contributors step up to do the releases every 6 weeks and so far it hasn't been a problem as the release process is fairly lightweight. Testing beyond unit and e2e tests is done by Weaveworks and Grafana Labs who run unreleased binaries in production.

- **Besides the core team, how active is the surrounding community? Bug reports? Assistance to newcomers? Blog posts etc.**

We have engineers from Rakuten, Rewe and other major end-users constantly helping in the slack channel and github via issues and PRs.

- **Do they make it easy to contribute to the project? If not, what are the main obstacles?**

I would say yes, see: https://cortexmetrics.io/docs/contributing/

- **Are there any especially difficult personalities to deal with? How is this done? Is it a problem?**

None we are aware of.

- **What is the rate of ongoing contributions to the project (typically in the form of merged commits).**

See * Commits per week over the last 6 months
* Issue Opened/Closed per week over the last 6 months
* New PRs per week over the last 1 year

2020-05-26: SIG o11y is happy with the answers in the above section

Users

- **Who uses the project? Get a few in-depth references from 2-4 of them who actually know and understand it.**

Grafana Labs and Weaveworks run massive commercial offerings on top of Cortex. See some case-studies of Cortex by end-users here: https://www.weave.works/blog/how-aspen-mesh-runs-cortex-in-production and https://cortexmetrics.io/docs/case-studies/gojek/

- **What do real users consider to be its strengths and weaknesses? Any concrete examples of these?**

See: https://docs.google.com/spreadsheets/d/1qZKhgIlxrsPHdSVgWqzT28rfuqhHrjL7aSMdVwmLyaA/edit#gid=1392778618 for anonymised survey results. Note: We had 18 responses from 17 unique companies. This survey doesn't include Weaveworks or Grafana Labs.

- **Perception vs Reality: Is there lots of buzz, but the software is flaky/untested/unused? Does it have a bad reputation for some flaw that has already been addressed?**

Can’t comment on the buzz, but there was some valid criticism in the past about Cortex being undocumented and hard-to-use/operate in the past which we have addressed over the past 18 months with our website and single-binary mode.

2020-05-26: SIG o11y is happy with the answers in the above section

Context

- **What is the origin and history of the project?**

Cortex started as a hosted Prometheus solution at Weaveworks to be part of Weave cloud before being donated to the CNCF. Freshtracks, Aspen Mesh and EA were some of the early adopters. Soon, Grafana Labs also started offering a hosted Prometheus solution on top of Cortex and started investing heavily in the project. The collective work of Weave, early adopters and Grafana Labs made Cortex scalable and fast and easy to use and increased adoption.

- **Where does it fit in the market and technical ecosystem?**

Cortex is part of a monitoring solution, and as such is complementary to technologies such as Kubernetes.

- **Is it growing or shrinking in that space? Is that space growing or shrinking?**

Cortex is definitely growing and we’re talking to more and more potential users every month. A significant portion of them move their POCs to production. Regarding the OSS monitoring space growing, I would say yes. More and more people are migrating from proprietary or old monitoring to Prometheus and are constantly looking for ways to scale and manage their deployments, with Cortex and Thanos as some of the top options.

- **How necessary is it? What do people who don't use this project do? Why exactly is that not adequate, and in what situations?**

Significant portion of the cloud-native world uses Prometheus for its monitoring, and it works well for most users. But it doesn’t have a few things out of the box:

1. Long-term replicated storage and scaling out: While you could run Prometheus in HA mode, if you lose a replica, it’s very hard to replicate the data again and it doesn’t guarantee no data loss. Further, scaling out a Prometheus deployment is not as trivial as adding more Prometheus nodes.

2. Multi-tenancy: Prometheus doesn’t support multi-tenancy.

If you need #1, you need a solution like Cortex or Thanos. If you need Multi-tenancy, so far Cortex is the only solution that supports Multi-tenancy out of the box but Thanos has plans to add multi-tenancy support.

- **Clearly compare and contrast with peers in this space. A summary matrix often helps. Beware of comparisons that are too superficial to be useful, or might have been manipulated so as to favor some projects over others. Most balanced comparisons will include both strengths and weaknesses, require significant detailed research, and usually there is no hands-down winner. Be suspicious if there appears to be one.**

Thanos is another CNCF project that provides high-availability and long-term storage to Prometheus. Both Thanos and Cortex make different trade-offs that will appeal to different use-cases.

- Cortex is a centralized store while one of the recommended deployments of Thanos holds the recent data at the edge in Prometheus servers themselves. This presents a different tradeoff — pushing writes to a central location with Cortex vs. pulling data at query time with Thanos. The tradeoff results in query latency and availability differences. Having said that, Thanos now supports an experimental receive service which can be used to centralise the data as well.

- Multitenancy is built into Cortex, which makes it a good option for larger organisations that need to keep the data from separate teams separate. Note: Thanos has multitenancy on the roadmap.

- Cortex leverages the built-in Prometheus remote-write API whereas the Thanos architecture allows for incremental adoption and reuse of existing Prometheus deployments.

- Cortex uses a lot of caching to speed up queries and we are actively working with the Thanos community to bring much of the same improvements to Thanos.

With all this, the Cortex and Thanos communities are constantly collaborating with each other, the recent ones being using the Cortex query-frontend to provide query caching for Thanos and exploration of writing blocks and using the Thanos query path in Cortex. There are plans to collaborate further and the community and usage of both the projects is only growing!

This is further aided by the large overlap of Prometheus, Cortex, and Thanos maintainership and code, and close coordination between all three projects.

CNCF being explicit about not wanting to be kingmakers, they encourage competition between projects. Furthermore, each project needs to pass due diligence on its own merits without considering other projects.

TL;DR:

- Cortex is growing
- Cortex collaborates with nearest competitor in CNCF space
- Cortex solves important problem

Thanos and Cortex started out with different trade-offs and assumptions, but as they evolved, they are collaborating and getting closer and closer.

As an end user deciding which to use, if you care most about scaling out storage, you should be looking at Thanos first. If you care about scaling the query performance, you should be looking at Cortex first. That being said, Cortex and Thanos are getting closer and closer and might end up merging at some point.

2020-05-26: SIG o11y is happy with the answers in the above section