Sylabs.io

Year-End White Paper

Singularity Containers for Compute-Driven Workloads in 2018: Results for the Community and Community-Motivated Roadmap

Table of Contents

Executive Summary

2018 has proven to be a remarkable year for Singularity. First, we take a look back at what has been accomplished by Sylabs and the Singularity Community this year. Then we provide a number of updates as to what you can expect in the future for Singularity – whether you're already a member of the Community or just an interested bystander. With this year drawing to a close, we thought this was the perfect time to be a little reflective and make known aspects of our roadmap, as we seized numerous opportunities to engage with broader communities on the progress and direction of this critical solution for containerizing HPC and numerous other compute-driven workloads.

A brief summary is as follows:

- Key accomplishments include reimplementation of the Singularity core in the Go programming language, along with introduction of the Singularity Image Format (SIF); together, these milestones factored significantly in the version 3.0 Community Edition release. Additionally, the Sylabs Cloud is a Singularity-centric portal hosted in the cloud that enables key signing and verification, hosted containers, and remote builds. Finally, Singularity was recognized by HPCwire readers and editors in receiving three awards at SC18.

- Under active development are native runtime interfaces for Singularity in Kubernetes and Nomad, compliance with standards emerging from within the Open Container Initiative (OCI), plus direct support for the Dockerfile syntax.

Looking Back: 2018 Accomplishments

In looking back, two strategic investments stand out in particular – investments that affected Singularity right to its very core, literally! Together with a Singularity-centric cloud portal, these milestones factored significantly in the version 3.0 Community Edition release. Finally, Singularity was recognized by HPCwire readers and editors in receiving three awards at SC18.

Singularity Code Reimplementation

The first strategic investment involved Sylabs' reimplementation of the Singularity codebase in the Go programming language. Motivated at the outset by the need to both modernize and unify the Singularity codebase into a single primary language, Go emerged as the obvious choice. While advantageous in and of itself, an example of that would be from the perspective of software lifecycle management, the choice of Go offered the potential for a significant strategic upside. Briefly, it is increasingly the case that Go is the go-to language for many of the major projects in the container ecosystem. Therefore, the investment in a Go-based refactoring seemed to be the most beneficial thing to do. As we'll see later on, strategic uptake of Go proves extremely important in the roadmap for Singularity in the immediate future. Finally because it remains an open source-based project, you can perform your own code-level review to see how this transition to Go has materialized starting with version 3.0 of Singularity.

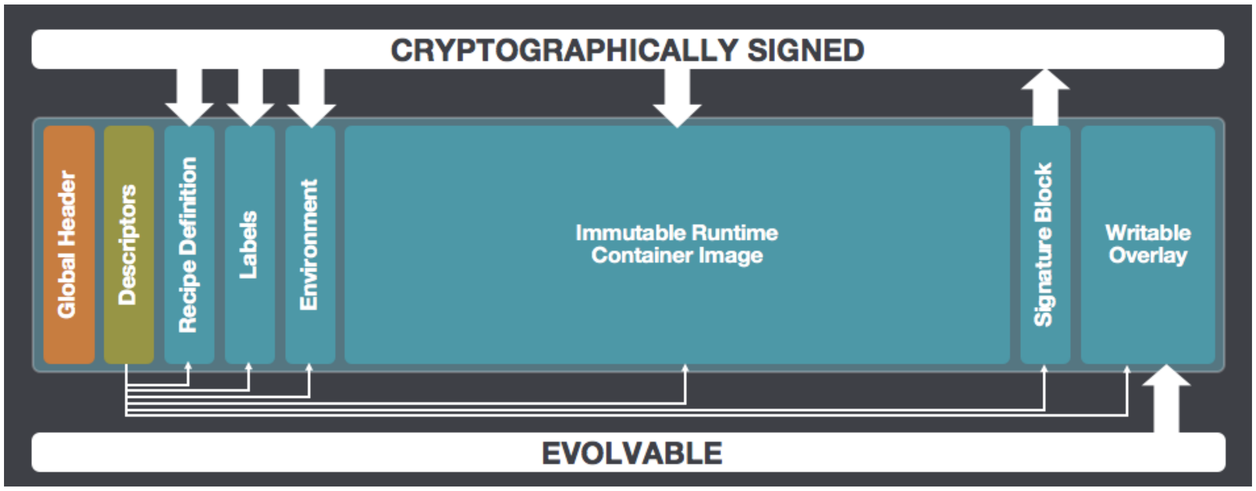
Go is the go-to language for many of the major projects in the container ecosystem. Therefore, a Go-based refactoring of the Singularity core seemed to be the most beneficial thing to do.

Singularity Image Format (SIF)

Encapsulating and abstracting any OCI compatible container format (or a build from scratch) as a single cryptographically signed and trusted file. What could be simpler than that? Literally! With the introduction of the Singularity Image Format (SIF), this strategic vision has been realized in a fashion that is as technically elegant as it is simple, and comprises Sylabs' second strategic investment of 2018. Boiling many aspects of containerization down to the manipulation of a single file comes with HUGE strategic advantages, as this is a well-worn path.

In abstracting a container as a single file, SIF is as technically elegant as it is simple, secure, and mobile.

Sylabs laid out the original vision for SIF in a single blog post (of course!) towards the end of the first quarter of 2018. Whereas working at the filesystem level may at first blush not appear to be the most glamorous of pursuits, you'll readily see for yourself from the schematic below that SIF is indeed a thing of beauty. Briefly, SIF presents architecturally as a composable format that is logically structured, extensible, and of course highly mobile. To ensure that mobility does not come at the expense of security, SIF incorporates the ability to cryptographically sign and verify containerized images – essentials in ensuring a bond of trust from container provenance to execution destinations, regardless of any tampering that may have occurred in transit. Singularity and security have been joined at the hip since the outset. A significant aspect of the rationale for developing SIF has always been to provide a foundation upon which Singularity can be embraced, enhanced, and extended in productive and innovative ways – with respect to security-related and other capabilities. Even before it was actually prototyped and implemented back in March 2018, the vision for SIF in the Looking Forward section.



SIF is not only logically structured, extensible, and highly mobile, it also includes the ability to cryptographically sign and verify containerized images.

Singularity 3.0

The use of SIF as the de facto standard format for Singularity images comprised a significant aspect of the rationale for a major-version increment to release 3.0 of this software in October 2018. Refactoring and reimplementation via Go, in tandem with the introduction of SIF as the default packaging mechanism for the Singularity container runtime, collectively produced a compelling software release. However, in addition to these changes to Singularity itself, efforts made to better enable the software's nascent ecosystem served as an additional focal point for the 3.0 release. Released alongside Singularity as an alpha-state preview, Sylabs Cloud is a Singularity-centric portal hosted in the cloud and comprised currently of the following three components:

- **Key Management Services** – a service that signs and validates SIF-embedded cryptographic keys in Singularity containers

- **Container Library Services** – a repository to manage, store, and share Singularity containers in public and/or private

- **Remote Builder Services** – an image-building enabler for Singularity containers that provides secure, isolated environments to 'mere mortal' users, all without the need for root access to that system

As you might expect, the Sylabs Cloud is also fully integrated with the Singularity 3.0 Command-Line Interface (CLI).

The Sylabs Cloud is a Singularity-centric portal hosted in the cloud that enables key signing and verification, hosted containers, and remote builds.

HPCwire Awards

Singularity won ***three*** HPCwire Awards at SC18.

The first ***and*** second awards combined for a sweep of the "Best HPC Programming Tool or Technology", as Singularity was the sole choice for *both* HPCwire readers as well as editors in this category. Tom Tabor, CEO of HPCwire's parent company and publisher Tabor Communications, related in person to us just how unusual such a category sweep was, with HPCwire readers and editors independently making their assessments. Singularity previously received accolades from HPCwire readers in winning this same award in 2017.

In winning the "Top 5 New Products or Technologies to Watch" this year, Singularity completed 'the three-peat' as HPCwire has bestowed this honor upon the software for three years in a row. What's significant with the 2018 recognition however, is that for the first time this was the readers' selection, whereas in 2016 and 2017 it was the HPCwire editors who were responsible for singling out Singularity. We believe this signals a very important shift regarding the maturity of the software.

From a single award in 2016, to two awards last year in 2017, to three this year, it is truly gratifying to witness the recognition Singularity *continues* to receive – recognition that identifies it as the the most appropriate way of containerizing the compute-driven workloads that characterize HPC and Enterprise Performance Computing (EPC).

Increasing recognition for Singularity aside, this year's awards were extremely special for at least two other **major milestones**: not only does SC18 mark the 30th anniversary of supercomputing's main event, this year also marked the 30th anniversary of HPCwire itself. Both of these are remarkable achievements that point to the staying power of technological innovation at the frontiers of scientific and other compute-driven disciplines.

Voted award-winning software for three years running by the readers and editors of HPCwire.

Three awards for Singularity. 30 years for SC and HPCwire. 2018 was all about the threes. We are humbled and delighted to be working with the open source community to deliver award-winning software at a truly significant time in the history of HPC.

Accolades aside, today Singularity is depended upon by more than 30,000 top academic, government, and enterprise users, who trust it to run millions of jobs each day on more than 3 million sockets. The Community's response regarding the introduction of SIF, adoption of Go in the Singularity core codebase, as well as the Sylabs Cloud ecosystem-enabling cloud portal, has been nothing short of resoundingly positive and encouraging. The strategic investments in Go and SIF, alongside the burgeoning Singularity ecosystem, positions the software exceedingly well for the multitude of projects that the Community has on its collective roadmap. Before delving into its specifics in earnest, however, a use case interlude provided by the Community illustrates how their interests continue to evolve Singularity in support of their increasingly demanding requirements.

Emerging Community-Driven Use Cases

The Sylabs Cloud has the potential to be a significant enabler of the Community's efforts with and about Singularity. To better ensure that this is the case in reality, a number of members of the Community have been expending effort on the development of use cases. As of this writing, there exists an impressive array of examples that contributors anticipate to seed the efforts of others in the Community. In some cases, these examples will serve as ends in themselves, as they provide ready-to-run solutions that implement Singularity containers to address a number of common needs spanning programming languages, data (from relational databases to message busses), and Internet services (e.g., HTTP server), graphics processing plus Internet of Things (IoT), and Machine Learning. Of course, individually containerized examples can be assembled into workflows. Finally, these examples provide tangible illustrations for Community members as to how easily they to can contribute use cases that they perceive to be of value to other members of the Singularity Community – if you can write a 'recipe' for a container via a Singularity definition file, you can contribute!

> Hybrid use cases combine streamed workloads, real- time analysis, and data pipelining into compute-focused services.

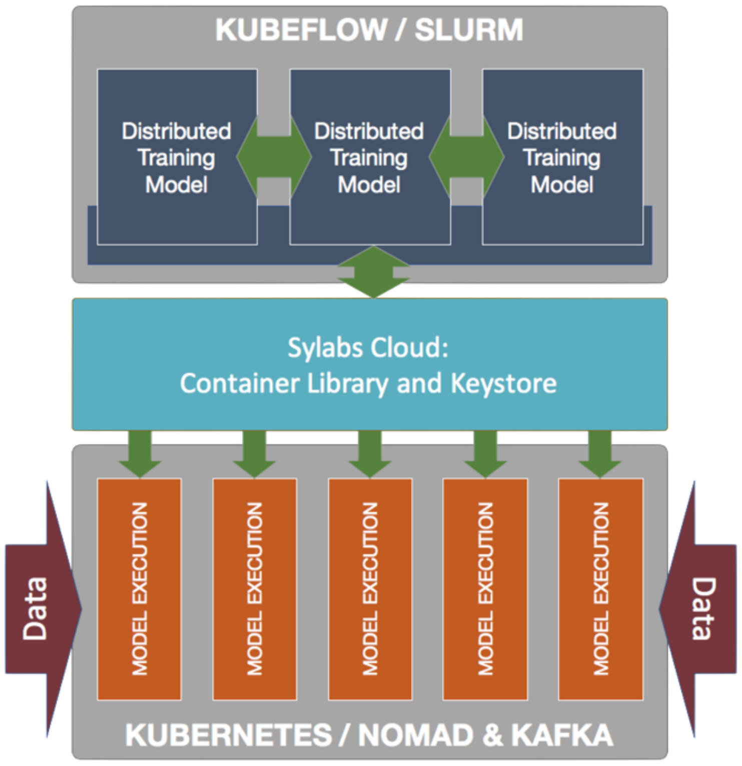
Whereas seeding uptake of Singularity containers for isolated applications comprises a compelling outcome for the collective efforts of the Community in the Sylabs Cloud, it's evident to us that the even more compelling real-time workflows are increasingly becoming the 'new normal.' The figure below presents an illustrative example based upon Machine Learning that we can deconstruct as follows:

- Training frequently emerges as one of the most computationally demanding workflow components in many applications of Machine Learning – often, ironically, as the harbinger of making the successful transition from prototype to production. To address this challenge, frameworks from Keras, PyTorch to TensorFlow employ distributed processing (native and/or via Horovod) to take advantage of parallelism in an effort to reduce the time required for training. If data is being streamed in real time, from social-media (e.g., Twitter) and/or other sources, there is an implicit requirement to train on an ongoing basis. In the figure below, distributed training is achieved through use of Google's Kubeflow toolkit for Machine Learning. The resulting training computations are distributed across a cluster through use of the Slurm workload manager.

- The Sylabs Cloud facilitates distributed training by hosting signed/verifiable and immutable SIF containers for workflows such as the Kubeflow/Slurm combination described above. Each instance of the distributed-processing workflow for training accesses the same SIF container. Because SIF encapsulates the container runtime into a single file, training is conducted in a highly repeatable fashion across an array of workload-managed compute nodes – thus delivering mobility without any loss of security. The Sylabs Cloud facilitates execution of the Machine Learning model in an analogous fashion – also by exploiting the runtime mobility of signed/ verifiable SIF containers.

- The exploitation of trained models is emphasized by the data-driven model execution illustrated schematically in the lower part of this figure. From algorithms to framework, the specifics of the Machine Learning implementation are less of concern here, as the use case is broadly applicable. Those currently employing this paradigm have adopted Nomad, for example, to deploy the application workflow in a Kubernetes container cluster. The need to manipulate data streamed in real time is addressed by Apache Kafka – a distributed streaming platform ideally suited to the demands of manipulating 6V Big Data in real time.

At this juncture, the Singularity Community has advanced a number of use case examples that all have the above paradigm in common – i.e., streamed workloads, real-time analysis, and data pipelining into compute-focused services. What may not be readily obvious from examples like this is the innate ability of Singularity to singularly bridge the containerization requirements of both the


KUBEFLOW / SLURM
Distributed Training Model
Distributed Training Model
Distributed Training Model
Sylabs Cloud:
Container Library and Keystore
Data
MODEL EXECUTION
MODEL EXECUTION
MODEL EXECUTION
MODEL EXECUTION
MODEL EXECUTION
Data
KUBERNETES / NOMAD & KAFKA

The Sylabs Cloud facilitates hybrid use cases requiring hosted and trusted Singularity containers for compute-centric distributed training plus replicated, data-driven model execution services style in real time.

compute-driven workloads that characterize Enterprise Performance Computing (EPC) simultaneously with those typical of services-based deployments. Whereas Community members are already employing Singularity containers in the demanding, complex use cases shared here, this uptake has resulted in various additional requirements. To ensure that Singularity continues to be relevant to its Community, in the final section we share items from the roadmap that better place the container technology in terms of relevance and usefulness in serving all its stakeholders.

Looking Forward: Roadmap Beyond 2018

The previous section concluded with a demanding and complex use case that straddled the realms of both EPC and services-based deployments. Whereas containerization via Singularity today affords the ability to entertain such use cases, broader and deeper adoption within the Community demands that additional requirements be addressed. A number of these requirements are covered briefly in this final section, with the promise that subsequent, more focused content will allow members of the Community to elaborate as appropriate. As possibly the most-encouraging takeaway we can share at the current time, all of these roadmap items are not only committed to, most are already well underway.

Standards Compliance

As alluded to much earlier in this paper, and with respect to the original vision for SIF, we noted that block-based encryption for any data object was in scope and on the roadmap. As we work to bring Singularity into full compliance with standards emerging from the Open Container Initiative (OCI), this data-encryption requirement is slated for immediate action. Also related to OCI compliance, and involving SIF specifically, will be the need to harmonize the existing signing and verification capability inherent in SIF with those requirements currently emerging from within the OCI. Because extensibility was designed into SIF from the outset, OCI compliance can be addressed through technically efficient and effective means. Finally, and related to compliance with respect to the OCI Image Specification, will be efforts to ensure that SIF images can be appropriately 'ingested' – e.g., for subsequent use in an OCI-compliant runtime. In other words, SIF becomes a packaging format for OCI-based containers.

We are working to bring Singularity into full compliance with standards emerging from the Open Container Initiative (OCI).

Still related to SIF, compliance will include the ability to mount an arbitrary number of SIF images as an OCI bundle. Needed for compliance with the OCI Runtime Specification, at a high level, this translates to mounting an arbitrary number of files – as SIF encapsulates each Singularity container into a single file. Of course, it'll be the dogged efforts of working the low-level details to ensure ultimate compliance with SIF for runtime purposes. Keeping with the OCI runtime, and not too surprisingly, efforts are also underway to draw Singularity's CLI and Application Programming Interface (API) into compliance.

The implementation of SIF in Singularity, as well as the Go-based core implementation that is Singularity itself, is all licensed under the three-clause BSD license available online at the project's GitHub site. Whereas it might be effectively argued that open source-based Singularity is the de facto standard for containerization implementation in HPC and EPC, it is not today fully compliant with open standards for containerization within the enterprise. As an active participant in the development of those standards that will ultimately define image specifications, runtimes, and more, in support of containerization, Sylabs and the Singularity Community is committed to the OCI process and outcomes as they will ultimately deliver benefits for all stakeholders in the entire containerization ecosystem.

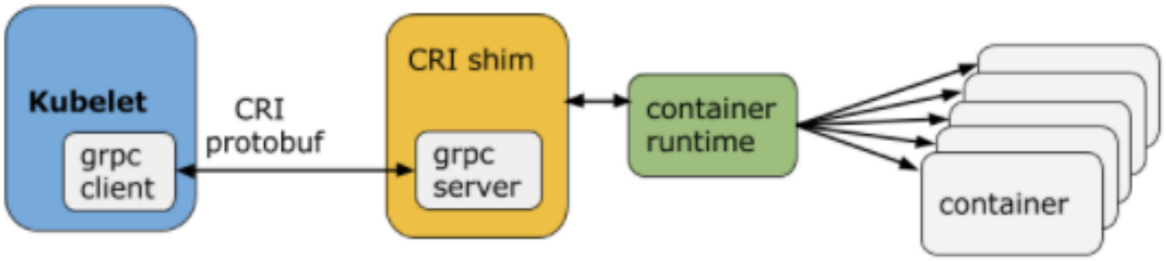
Kubernetes Integration

As the schematic presented as a motivating use case indicates, the Community's interests in adopting Kubernetes for orchestrating Singularity containers spans both services-based deployment models as well as (increasingly) those originating from HPC and EPC. Whereas the former comprises a canonical example for Kubernetes uptake, the latter might be received with a degree of surprise. Recall however, the high-level description of the use case – namely streaming workloads, real-time analysis, and data pipelining into compute-focused services. When taken en masse, Kubernetes rapidly emerges as a better fit than pre-existing capabilities (e.g., workload management) in EPC or even HPC.

With the Community's need then originating from use cases such as the compute-services hybrid shared above, it's not too surprising that tightly coupled integration and interoperability with Kubernetes has emerged as a pressing requirement for Singularity. The need to be immediately addressed here relates to integration between Singularity containers and the Kubernetes runtime. Fortunately, as of version 1.5 of Kubernetes in 2016, a Container Runtime Interface (CRI) was released. Improved in two subsequent 'dot' releases of

The Kubernetes integration effort is already underway via a new open source project. The Container Runtime Interface (CRI) in Kubernetes for Singularity includes image and runtime services.

Kubernetes, Kubelet interaction with this CRI is illustrated schematically below. Evident from this figure is that the node-persistent agent known as a “Kubelet” is now able to work with any container runtime that provides a gRPC server implementing the CRI. Thus immediately upon our agenda is the need to implement the Kubernetes CRI ‘shim’ (see schematic) to allow Singularity containers to be orchestrated by Kubernetes.



Kubelets are able to work with container runtimes that provide a gRPC server implementing the Kubernetes CRI, enabling Singularity image and runtime integration.

[Image credit: Yu-Ju Hong, Software Engineer, Google
Introducing Container Runtime Interface (CRI) in Kubernetes
https://kubernetes.io/blog/2016/12/container-runtime-interface-cri-in-kubernetes/]

Basically, the Singularity CRI implementation is a gRPC server which serves Kubelet requests by interoperating with the Singularity runtime. This server will interact with the Kubernetes CRI to natively spawn Singularity containers, and add SIF support to Kubernetes. As of this writing, the Kubernetes integration effort is already well underway via a new open source project. This proof-of-concept implementation is the above pictured “CRI shim.” Specifically, the Singularity CRI incorporates two services:

- **ImageService** – an interface responsible for pulling, removing, and maintaining the list of local Singularity images
- **RuntimeService** – an interface responsible for creating, terminating and otherwise managing Singularity containers within Kubernetes pods

Based upon the compute-services hybrid use case we shared earlier, you can well appreciate the enthusiasm with which the pursuit of integration with Kubernetes is being both pursued and received within the Singularity Community and beyond – let's face it, it's not every day that entirely new workflow patterns are enabled for complex use cases that span EPC and services-oriented computing.

Dockerfile Syntax Support

As of the version 3.0 release, not only can Singularity still make use of existing Docker images from the Docker Hub repository, it can reformat them into SIF. Once reformatted into SIF, of course, the reliance upon the source Docker image is removed, and the image can be managed as can any other SIF file – as the single means for encapsulating the runtime corresponding to a Singularity container.

> Interoperability with Docker has always been a strength of Singularity. Direct support for the Dockerfile syntax is on our roadmap.

Owing to the predominance of Docker in architecting microservices based applications, and the unsurprising and significant investment made by those making use of the technology including repository-hosted images, interoperability with Docker has always been a strength of Singularity. In broadening and deepening that interoperability, support for Docker in Singularity is about to be further enhanced through the addition of direct support for the Dockerfile syntax. Paring down Docker images to a build 'recipe' captured through a single text file, Dockerfiles are representative of the most-efficient means for detailing containerized images – Singularity, of course, employs definition files for this purpose. When you review definition files, such as those rapidly appearing in the examples area of the project's GitHub site, you'll quickly appreciate just how valuable direct support for the Dockerfile syntax has the potential to be. Furthermore, owing to the not always subtle implementation differences between Docker and Singularity, by working at the build level it is expected that higher degrees of compatibility will be ultimately ensured.

Nomad Support

As noted in the section that provided the motivating use case for our roadmap, Nomad is already factoring into some leading-edge use case examples. To reiterate here, Nomad is being used to deploy application workflows in Kubernetes container clusters. From hardened use case pioneers to those wanting to employ Nomad for the first time, it is clear that there is value in integrating the Nomad runtime with Singularity directly; thus, through use of a (Nomad) plugin, Singularity will be enabled as a container runtime under Nomad.

From native runtime interfaces for Singularity in Kubernetes and Nomad, to OCI standards compliance plus direct support for the Dockerfile syntax, there are a number of projects underway at the present time. Owing to the anticipated release of these latest efforts by and for the Singularity Community, in tandem with the potential to rapidly accelerate the implementation of game-changing compute-services hybrid use cases, realization of these roadmap projects will deliver immediately leverageable implementations while addressing important strategic objectives.

Through use of a plugin, Singularity will be enabled as a container runtime under Nomad.

Call to Action

As outlined in a recent blog post, SC18 provided numerous opportunities to engage formally or informally with Singularity and its rapidly growing Community – from BoFs and a panel in the Technical Program to presentations, hands-on workshops, and scavenger hunts on the exhibits floor. Because there were discussions as to current challenges and opportunities within the container ecosystem, here we've made efforts to ensure you are aware of the progress that has been made in 2018, and what you can expect from all of us over the next few weeks, months, and quarters. As of this writing, award-winning Singularity has progressed significantly, and established a vibrant and active Community; however, there is always room for additional Community members and a variety of ways in which contributions can be made. The Singularity Community looks forward to the ongoing dialog, as together we can continue to develop this unique approach for containerization into an even better solution for your existing and future use cases in HPC, EPC, and more.

There is always room for additional Community members and a variety of ways in which contributions can be made.

To learn more, please visit:

https://www.sylabs.io