INTEGRACIÓN DE APLICACIONES DESARROLLADAS CON MALLAS DE MICROSERVICIOS EN LA NUBE MEDIANTE LA PLATAFORMA DAPR

AGUDELO OSORIO DANIELA

UNIVERSIDAD DE MANIZALES

UNIVERSIDAD DE MANIZALES
FACULTAD DE CIENCIAS E INGENIERÍA
INGIENIERÍA DE SISTEMAS Y TELECOMUNICACIONES
MANIZALES
2021

INTEGRACIÓN DE APLICACIONES DESARROLLADAS CON MALLAS DE MICROSERVICIOS EN LA NUBE MEDIANTE LA PLATAFORMA DAPR

AGUDELO OSORIO DANIELA

Trabajo de Grado presentado como opción parcial para optar
al título de Ingeniera de Sistemas y telecomunicaciones

UNIVERSIDAD DE MANIZALES
FACULTAD DE CIENCIAS E INGENIERÍA
PROGRAMA
MANIZALES
2021

CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

LISTA DE CUADROS

Pág.

GLOSARIO

- **API Dapr**: son las API HTTP RESTful definidas por Dapr que los desarrolladores interactúan con las aplicaciones de construcción, entre las que se enumeran:

    - Referencia de la API de invocación de servicio
    - Referencia de la API de administración de estado
    - Referencia de la API de Pub / subReferencia de la API de enlaces
    - Referencia de la API de actores
    - Referencia de la API de secretos
    - Referencia de la API de salud
    - Referencia de la API de metadatos
    - Códigos de error devueltos por las API (Microsoft, 2021)

- **API de Azure:** Azure API Management proporciona una API REST para realizar operaciones en entidades seleccionadas, como usuarios, grupos, productos y suscripciones. Esta referencia proporciona una guía para trabajar con la API REST de administración de API, así como información de referencia específica para cada operación disponible, agrupada por entidad. (Microsoft, 2016)

- **API en APIM:** plataforma de gestión híbrida de múltiples nubes para API en todos los entornos, ayudando a las organizaciones de todos los tamaños a diseñar, proteger, publicar, escalar y observar las API en las nubes y en las instalaciones y hacer que los desarrolladores internos, socios y públicos puedan descubrirlas y consumirlas. (Microsoft Azure, 2017)

- **Directivas de Azure API Management:** Como lo define Microsoft (2017), las directivas son una colección de declaraciones que se ejecutan secuencialmente en la solicitud o respuesta de una API. Entre las declaraciones más usadas se encuentran la conversión de formato de XML a JSON y la limitación de tasa de llamadas para restringir la cantidad de llamadas entrantes de un desarrollador. Hay muchas más directivas disponibles y listas para usar. Estas, se aplican en la puerta de enlace que se encuentra entre el consumidor de la API y la API administrada.

- **Microservicio:** según Amazon (2019), los microservicios son un enfoque arquitectónico y organizativo para el desarrollo de software, donde el este se compone de pequeños servicios independientes que se comunican a través de API bien definidas y que permiten a las aplicaciones ser más fáciles de escalar y más rápidas de desarrollar.

RESUMEN

Sin lugar a dudas, el desarrollo de aplicaciones distribuidas ha sido un desafío para la comunidad de ingenieros en todo el mundo, gracias a DAPR, se abordan estos temas de forma más simple, a través del uso de una arquitectura de componentes conectables y simplificando de manera sustancial la estructura detrás de las aplicaciones distribuidas, pues proporciona un pegado dinámico que enlaza la aplicación con las funcionalidades de infraestructura del tiempo de ejecución de DAPR.

Con el fin de conocer de cerca estas ventajas que Dapr ofrece a los desarrolladores, se pretende documentar para la fábrica de software de la Universidad de Manizales, el proceso que debe abordar para ejecutar este proceso y evidenciar las características que se buscan fortalecer mediante la adopción de esta infraestructura en los propios desarrollos, permitiendo que para futuras implementaciones se puedan aplicar mejores prácticas e infraestructuras que potencien en su máxima expresión la intención de cada uno de los proyectos que se inicien.

PALABRAS CLAVES: Dapr, desarrollo, microservicios, api, .Net Core, Nodejs

ABSTRACT

Undoubtedly, the development of distributed applications has been a challenge for the engineering community around the world, thanks to DAPR, these issues are addressed in a simpler way, by using a plug-in component architecture and substantially simplifying the structure behind distributed applications, as it provides a dynamic paste that links the application with the infrastructure functionalities of the DAPR runtime.

To know these advantages that Dapr offers to developers, it is intended to document for the software factory of the University of Manizales, the process that must be approached to execute this process and demonstrate the characteristics that are sought to be strengthened through adoption of this infrastructure in the developments themselves, allowing future implementations to apply the best practices and infrastructures that enhance the intention of each of the projects that are launched to its maximum expression.

KEY WORDS: Dapr, development, microservices, API, .Net Core, Nodejs

INTRODUCCIÓN

Cuando en términos de ingeniería se habla sobre el desarrollo de aplicaciones distribuidas de alto rendimiento, escalables y confiables, sin lugar a dudas se abordan temas densos y de bajo nivel en procesos de cómputo, debido a que se deben considerar muchos conceptos de arquitectura, uso de contenedores, diversos lenguajes de programación, entre otros.

En respuesta a esta necesidad, surgió la plataforma Dapr, ofreciendo patrones y prácticas comprobadas de desarrollo, unificando la semántica impulsada por eventos y actores en un modelo de programación simple y consistente. Adicionalmente, fue diseñada para ser compatible con todos los lenguajes de programación. No está expuesto a primitivas de bajo nivel como subprocesos, control de simultaneidad, particiones y escalado, dando lugar a la posibilidad de escribir código implementando un servidor web simple y utilizando marcos web familiares de fácil elección.

De esta manera, Dapr se ha convertido en una de las mejores opciones al momento de diseñar una aplicación nativa, pues es flexible en modelos de consistencia de estado y permite el aprovechamiento de subprocesos múltiples y la implementación de escenarios avanzados sin restricciones artificiales, por lo que se mostrará en el desarrollo del documento, el proceso de construcción de una aplicación de baja complejidad, basada en microservicios y utilizando esta plataforma para integrar dos lenguajes de programación que por definición no están diseñados para ser directamente compatibles, Python y NodeJs.

1. ÁREA PROBLEMÁTICA

Durante la última década, el desarrollo de sistemas y aplicaciones para brindar solución a diversas problemáticas y necesidades de la comunidad en general se ha convertido en una de las labores más demandadas y apetecidas dentro del mercado internacional, sin embargo, su implementación en solicitudes de alta complejidad no es un proceso que se ejecute de forma ágil y eficiente, pues exige por parte de sus profesionales, un amplio conocimiento en arquitectura para aplicaciones distribuidas y otros aspectos de la ingeniería de sistemas que permitan su migración e implementación en diferentes entornos, haciendo que los procesos sean demorados, costosos y en ocasiones ineficientes en su concepto de arquitectura.

En la fábrica de software de la Facultad de Ciencias e ingeniería de la Universidad de Manizales, se hace necesario definir un estándar que contenga las mejores prácticas y patrones de desarrollo del mercado actual, que vayan a la vanguardia y que mantengan actualizados los conceptos tecnológicos que abordan las grandes compañías del mundo, como lo es Microsoft y su nuevo lanzamiento Dapr, siendo una plataforma de ejecución simplificada para el desarrollo de aplicaciones nativas de la nube y que le permite a los desarrolladores dedicarse fundamentalmente a lo importante de su ejercicio, concentrarse en la lógica central de su aplicación y mantener su código simple y portátil, reducir el umbral para la innovación en programación nativa y moderna y propender el uso arquitectura de microservicios, mediante patrones que admitan varios lenguajes de programación y, que por lo tanto, conlleven a que las aplicaciones sean más livianas con su tiempo de ejecución, convirtiéndose en una opción especialmente útil al migrar o implementar un producto en diferentes entornos, pues en otras circunstancias, se transformaría en un proceso que puede ser muy cerrado y en algún momento complicado.

Por lo anterior, la existencia de un documento con bases sólidas y que evidencia todo el potencial que tiene Dapr como plataforma neutral y libre de proveedores, permitirá a los estudiantes cque realizan desarrollos dentro de la fábrica de software, garantizar altos niveles de calidad, uso de tecnologías de punta, revolucionarias, de código abierto y que contribuyan a la formación de excelentes profesionales en todos los campos de la ingeniería.

2. OBJETIVOS

2.1 OBJETIVO GENERAL

Documentar e identificar el proceso de desarrollo de una aplicación de baja complejidad con arquitectura de microservicios, mediante el uso de la plataforma DAPR como herramienta de soporte, permitiendo identificar así, las mejores prácticas en desarrollos futuros para la Fábrica de software propia de la Universidad de Manizales.

2.2 OBJETIVOS ESPECÍFICOS

- Preparar los entornos de trabajo pertinentes para el desarrollo de una aplicación nativa que utilice la plataforma Dapr como entorno de ejecución en tiempo real.

- Desarrollar una aplicación nativa en la nube de baja complejidad, mediante el uso de una malla de servicios desarrollados con lenguaje Python.

- Desarrollar una aplicación nativa en la nube de baja complejidad, mediante el uso de una malla de servicios desarrollados con lenguaje NodeJs.

- Establecer una integración entre aplicaciones web utilizando la plataforma Dapr, que permita la comunicación de una malla de servicios desarrollados con los lenguajes NodeJs y Python.

- Establecer un benchmark de rendimiento de invocación de servicios y utilización de recursos de la plataforma DAPR en diferentes entornos de alojamiento para conocer su capacidad en ejecución.

3. JUSTIFICACIÓN

Sin lugar a dudas, las arquitecturas basadas en microservicios han surgido con gran notoriedad en los últimos tiempos entre los desarrollos en la nube gracias a las ventajas que se evidencian, como la mantenibilidad y la posibilidad de escalar servicios de forma independiente, de acuerdo a la demanda concreta que tenga cada uno de ellos; estas ventajas, entre otras, podrían resultar beneficiosas para abordar requerimientos de Plataformas de Integración. Sin embargo, también se ha señalado que la adopción de este tipo de arquitectura no es una solución general aplicable a todos los casos ya que posee complejidades y problemáticas intrínsecas a sus propias características, reconociendo así, que este es un proceso de alta complejidad si se ponen a consideración las diferentes variables de contexto, pues implica establecer una arquitectura de componentes conectables y en ocasiones con tecnologías que no están diseñadas para tal fin.

Dapr como solución, ofrece patrones de desarrollo y buenas prácticas para construir microservicios en "bloques de construcción" que son independientes, abiertos y exponen una API para una fácil integración y una clara separación de preocupaciones. Los desarrolladores simplemente usan un subconjunto deseado de estos bloques, construyendo de forma incremental para poner su aplicación en funcionamiento en un tiempo récord además de permitir a los desarrolladores que usen cualquier lenguaje o marco para escribir aplicaciones distribuidas, de tal forma que casi se encarga de resolver los problemas difíciles que al crear aplicaciones de microservicios que como se conoce en el mercado, gracias a su sencilla escalabilidad, se considera especialmente adecuado cuando se tiene que procurar la compatibilidad con un amplio sector de diferentes plataformas ya sea IoT, web, móvil, entre otras, o simplemente cuando no se conoce a ciencia cierta hacia qué tipo de dispositivos estamos orientando nuestro trabajo, proporcionando bloques de construcción más óptimos y eficientes.

4. MARCO TEÓRICO

En este capítulo se describen los aspectos teóricos necesarios para abordar de manera correcta los objetivos del proyecto, para lo cual se definen lo que es Dapr y sus características, metas en el mercado, componentes y algunas configuraciones técnicas importantes que se deben tener en cuenta.

4.1 Qué es Dapr

La mejor definición de Dapr la muestra Microsoft en su página oficial:

> *"Dapr es un tiempo de ejecución portátil impulsado por eventos que facilita a cualquier desarrollador la creación de aplicaciones resistentes, sin estado y con estado que se ejecutan en la nube y en el borde y que abarca la diversidad de lenguajes y marcos de desarrollo.*
> *Su principal misión es un desarrollo en cualquier idioma, cualquier marco, en cualquier lugar.*
>
> *Dapr se encarga de codificar las mejores prácticas para construir aplicaciones de microservicio en bloques de construcción abiertos e independientes que le permiten construir aplicaciones portátiles con el lenguaje y el marco de su elección. Cada bloque de construcción es completamente independiente y puede usar uno, algunos o todos en su aplicación.*
>
> *Además, Dapr es independiente de la plataforma, lo que significa que puede ejecutar sus aplicaciones localmente, en cualquier clúster de Kubernetes y otros entornos de alojamiento con los que se integra Dapr. Esto le permite crear aplicaciones de microservicio que se pueden ejecutar en la nube y en el borde.*
>
> *Con Dapr, puede crear fácilmente aplicaciones de microservicio utilizando cualquier lenguaje, cualquier marco y ejecutarlas en cualquier lugar".*

4.2 Bloques de construcción de microservicios para la nube y el borde

Dapr ofrece la posibilidad de diseñar aplicaciones de microservicios de manera estándar e implementarla en cualquier entorno. Lo hace proporcionando bloques de construcción de sistemas distribuidos que son independientes, lo que significa que puede usar uno, algunos o todos en su aplicación. Como se observa el cuadro 1, se proporcionan los componentes básicos de la misma:

Cuadro 1. Bloques de construcción de microservicios Dapr

BLOQUE DE CONSTRUCCIÓN	DESCRIPCIÓN
Invocación de servicio a servicio	La invocación resiliente de servicio a servicio permite llamadas a métodos, incluidos reintentos, en servicios remotos dondequiera que se encuentren en el entorno de alojamiento admitido.
Administración del Estado	Con la administración de estado para almacenar pares clave / valor, los servicios con estado de larga ejecución y alta disponibilidad se pueden escribir fácilmente junto con los servicios sin estado en su aplicación. El almacén estatal es conectable y puede incluir Azure CosmosDB, Azure SQL Server, PostgreSQL, AWS DynamoDB o Redis, entre otros.
Publicar y suscribirte	Publicar eventos y suscribirse a temas
Vinculaciones de recursos	Los enlaces de recursos con desencadenadores se basan en arquitecturas impulsadas por eventos para la escala y la resistencia al recibir y enviar eventos desde y hacia cualquier fuente externa, como bases de datos, colas, sistemas de archivos, etc.
Actores	Un patrón para objetos con y sin estado que simplifican la simultaneidad con el método y la encapsulación de estados. Dapr proporciona muchas capacidades en el tiempo de ejecución de su actor, incluida la concurrencia, el estado, la gestión del ciclo de vida para la activación / desactivación del actor y los temporizadores y recordatorios para los actores de activación.
Observabilidad	Dapr emite métricas, registros y seguimientos para depurar y supervisar tanto las aplicaciones de Dapr como las de los usuarios. Dapr admite el rastreo distribuido para diagnosticar y atender fácilmente llamadas entre servicios en producción utilizando el estándar W3C Trace Context y Open Telemetry para enviar a diferentes herramientas de monitoreo.
Misterios	Dapr proporciona administración de secretos y se integra con la nube pública y las tiendas de secretos locales para recuperar los secretos y usarlos en el código de la aplicación.

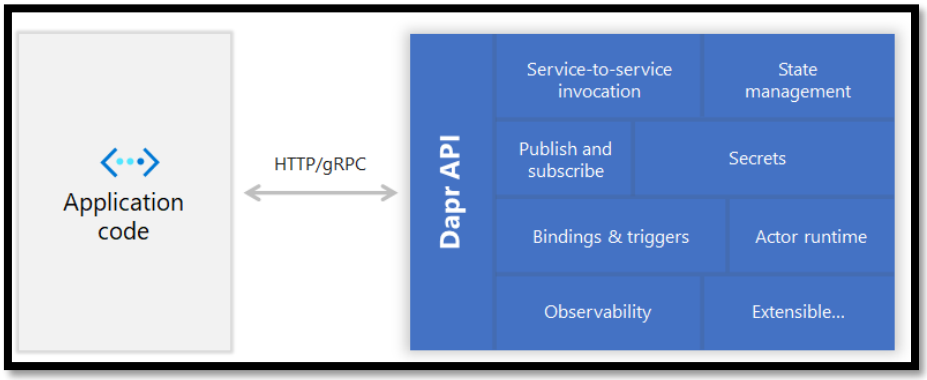
FUENTE: Dapr Docs. 2021. Overview. [online] Available at: <https://docs.dapr.io/concepts/overview/> [Accessed 22 March 2021].

Entornos de hospedaje

Microsoft (2019) en su última versión, muestra como Dapr se puede alojar en varios entornos, incluido el autohospedado para el desarrollo local o para implementarlo en un grupo de máquinas virtuales, Kubernetes y entornos de borde como Azure IoT Edge.

- **Auto alojado:** en la figura 1 se observa el modo autohospedado, Dapr se ejecuta como un proceso de sidecar separado al que su código de servicio puede llamar a través de HTTP o gRPC. En el modo autohospedado, también puede implementar Dapr en un conjunto de máquinas virtuales.

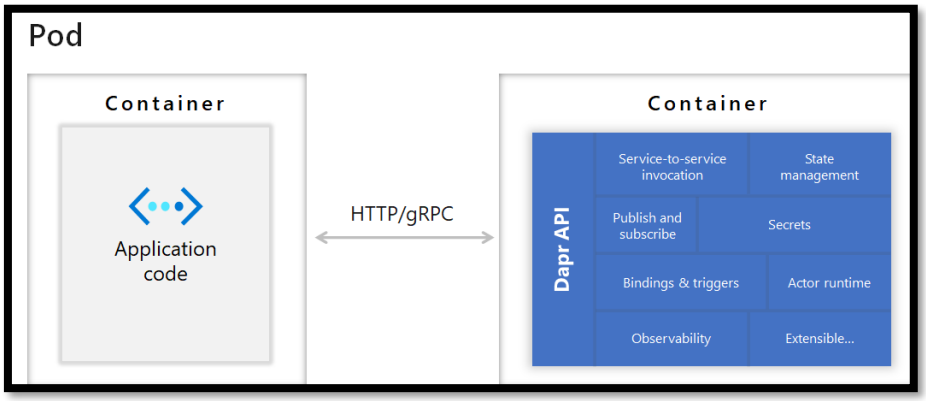
Figura 1 Entorno de hospedaje autoalojado



FUENTE: Dapr Docs. 2021. Overview. [online] Available at: <https://docs.dapr.io/concepts/overview/> [Accessed 22 March 2021].

- **Kubernetes alojado:** En entornos de alojamiento de contenedores como Kubernetes, Dapr se ejecuta como un contenedor lateral con el contenedor de la aplicación en el mismo pod, como se puede ver en la figura 2.

Figura 2 Entorno de Kubernetes alojado



FUENTE: Dapr Docs. 2021. Overview. [online] Available at: <https://docs.dapr.io/concepts/overview/> [Accessed 22 March 2021].

Frameworks para desarrolladores

Dapr se puede utilizar desde cualquier marco de desarrollo. A continuación, se muestran algunos que se han integrado con Dapr.

Web

En el SDK de Dapr .NET puede encontrar la integración ASP.NET Core , que ofrece controladores de enrutamiento con estado que responden a eventos pub / sub de otros servicios.

- *En el SDK de Dapr Java puede encontrar la integración de Spring Boot.*
- *Dapr se integra fácilmente con Python Flask y node Express.*
- *En Dapr PHP-SDK puede servir con Apache, Nginx o Caddyserver.*

Actores

Los SDK de Dapr admiten actores virtuales que son objetos con estado que simplifican la simultaneidad, tienen encapsulación de métodos y estados y están diseñados para aplicaciones distribuidas escalables.[1]

Funciones de Azure

Dapr se integra con el tiempo de ejecución de Azure Functions a través de una extensión que permite que una función interactúe sin problemas con Dapr. Azure Functions proporciona un modelo de programación basado en eventos y Dapr proporciona bloques de creación nativos de la nube. Con esta extensión, puede unir ambos para aplicaciones sin servidor y controladas por eventos. Para obtener más información, lea la extensión de Azure Functions para Dapr y visite el repositorio de la extensión de Azure Functions para probar los ejemplos, (Microsft Azure, 2019)

Flujos de trabajo de Dapr

Para permitir a los desarrolladores crear fácilmente aplicaciones de flujo de trabajo que utilicen las capacidades de Dapr, incluido el diagnóstico y la compatibilidad con varios idiomas, puede utilizar los flujos de trabajo de Dapr. Dapr se integra con motores de flujo de trabajo como Logic Apps. Para obtener más información, lea los flujos de trabajo nativos de la nube utilizando Dapr y Logic Apps y visite el repositorio de flujo de trabajo de Dapr para probar las muestras.

[1] Dapr Docs. 2021. *Overview*. [online] Available at: <https://docs.dapr.io/concepts/overview/> [Accessed 23 March 2021].

Diseñado para operaciones

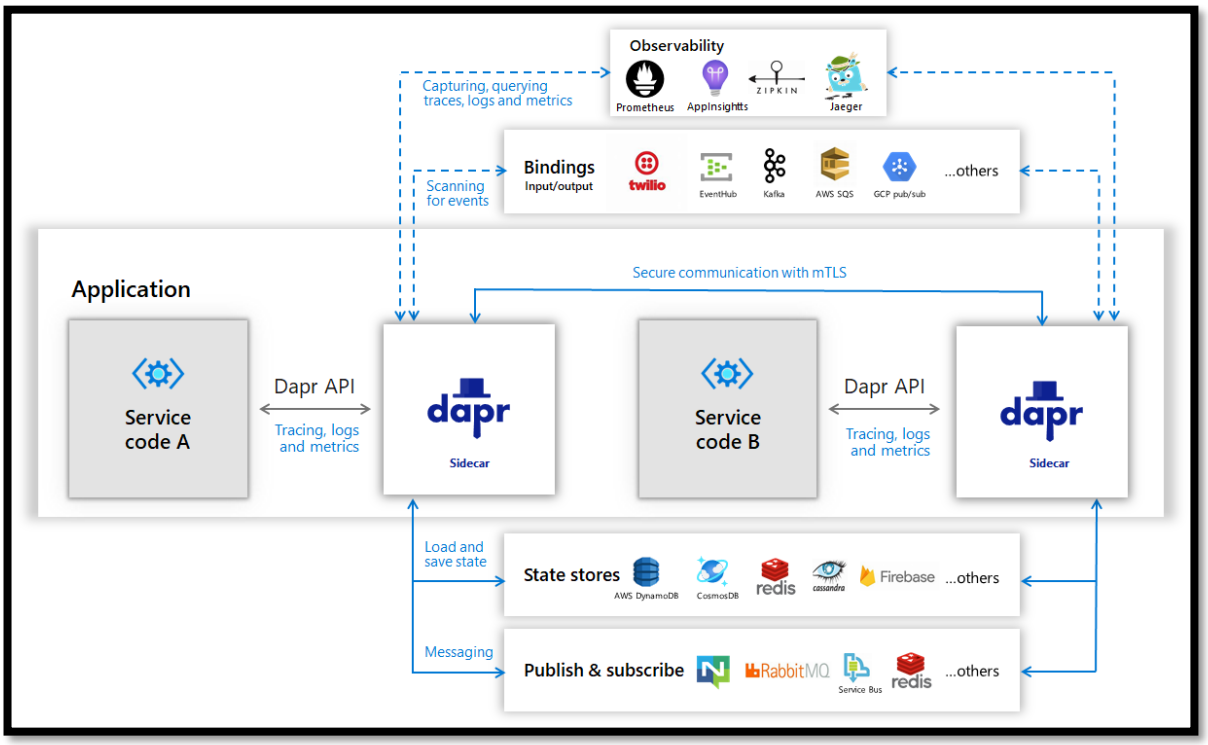
Según Microsoft (2020), Dapr está diseñado para operaciones. El panel de servicios, instalado a través de la CLI de Dapr, proporciona una interfaz de usuario basada en web que le permite ver información, ver registros y más para los sidecars de Dapr.

El soporte de las herramientas de monitoreo proporciona una visibilidad más profunda de los servicios y sidecar del sistema Dapr, y las capacidades de observabilidad de Dapr brindan información sobre su aplicación, como el seguimiento y las métricas.

Ejecutar Dapr en una máquina de desarrollador local en modo autohospedado

Como lo manifiesta Microsoft (2019), Dapr puede configurarse para ejecutarse en su máquina de desarrollador local en modo autohospedado. Cada servicio en ejecución tiene un proceso de tiempo de ejecución de Dapr (o sidecar) que está configurado para usar almacenes de estado, pub / sub, componentes de enlace y otros bloques de construcción, tal como se observa en la figura 3.

Figura 3 Arquitectura modo local autohospedado



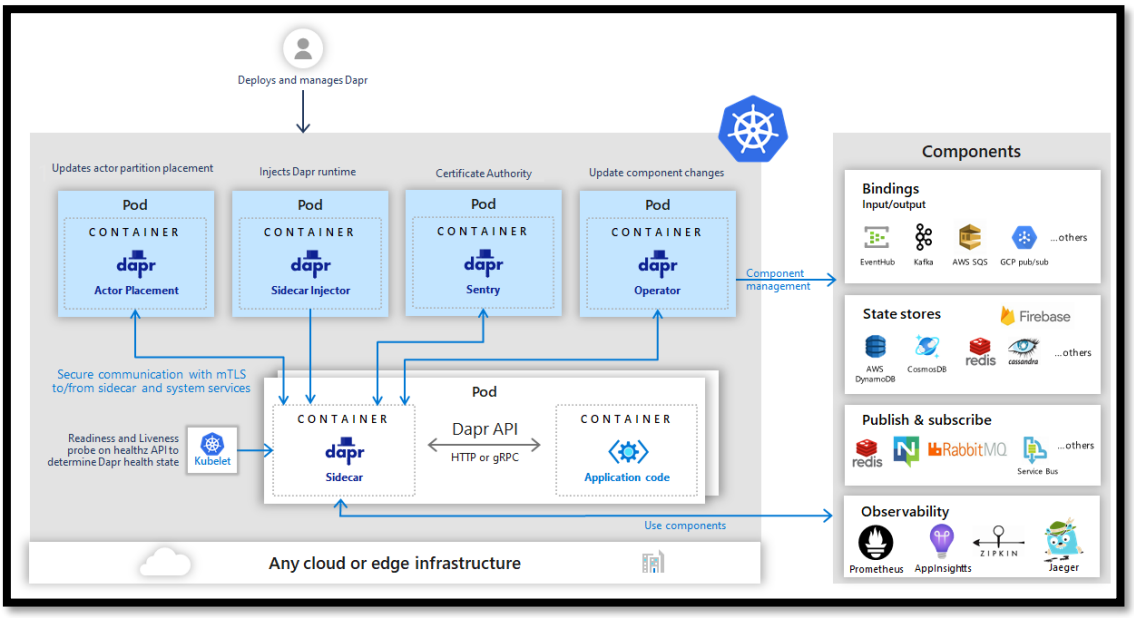
FUENTE: Dapr Docs. 2021. Overview. [online] Available at: <https://docs.dapr.io/concepts/overview/> [Accessed 22 March 2021].

Ejecutar Dapr en modo Kubernetes

Dapr se puede configurar para ejecutarse en cualquier clúster de Kubernetes. En Kubernetes, los servicios dapr-sidecar-injectory dapr-operator brindan una integración de primera clase para lanzar Dapr como un contenedor adicional en el mismo pod que el contenedor de servicios y proporcionar notificaciones de las actualizaciones de componentes de Dapr aprovisionadas en el clúster.

El dapr-sentryservicio es una autoridad de certificación que habilita TLS mutua entre instancias de sidecar de Dapr para el cifrado de datos seguro.

Figura 4 Implementación Dapr en modo Kubernetes



FUENTE: Dapr Docs. 2021. Overview. [online] Available at: <https://docs.dapr.io/concepts/overview/> [Accessed 22 March 2021].

4.3 Componentes

De acuerdo a la arquitectura definida por Microsoft (2020), Dapr utiliza un diseño modular donde la funcionalidad modular utilizada por bloques de construcción y aplicaciones Dapr utiliza un diseño modular donde la funcionalidad se entrega como un componente. Cada componente tiene una definición de interfaz. Todos los componentes se pueden conectar para que pueda cambiar un componente con la misma interfaz por otro. El repositorio de contribución de componentes es donde puede contribuir con implementaciones para las interfaces de componentes y amplía las capacidades de Dapr.

Un bloque de construcción puede utilizar cualquier combinación de componentes. Por ejemplo, el bloque de construcción de los actores y el bloque de construcción de la gestión estatal utilizan componentes de estado. Como otro ejemplo, el Pub / Sub edificio utiliza bloques componentes Pub / Sub.

Los siguientes son los tipos de componentes proporcionados por Dapr:

- Bindings
- Pub/sub
- Middleware
- Service discovery name resolution
- Secret stores
- State

Componentes de invocación y descubrimiento de servicios: Los componentes de descubrimiento de servicios se utilizan con el bloque de construcción de invocación de servicios para integrarse con el entorno de alojamiento para proporcionar descubrimiento de servicio a servicio. (Microsoft, 2021).

Invocación de servicios y componentes de middleware: Dapr permite que el middleware personalizado se conecte a la canalización de procesamiento de solicitudes. El middleware puede realizar acciones adicionales en una solicitud, como autenticación, cifrado y transformación de mensajes antes de que la solicitud se enrute al código de usuario o antes de que la solicitud se devuelva al cliente.

Componentes de la tienda secreta: En Dapr, un secreto es cualquier información privada que desee proteger contra usuarios no deseados. Las tiendas de secretos, que se utilizan para almacenar secretos, son componentes de Dapr y pueden ser utilizadas por cualquiera de los componentes básicos (Microsoft, 2021).

4.4 Seguridad

Uno de los mecanismos de seguridad que emplea Dapr para cifrar los datos en tránsito es la autenticación mutua TLS o mTLS, que ofrece algunas características clave para el tráfico de red dentro de su aplicación:

- Autenticación bidireccional: el cliente demuestra su identidad al servidor y viceversa
- Un canal cifrado para todas las comunicaciones en vuelo, después de que se establece la autenticación bidireccional

El TLS mutuo es útil en casi todos los escenarios, pero especialmente para sistemas sujetos a regulaciones como HIPAA y PCI.

Comunicación de sidecar a aplicación

El sidecar de Dapr se ejecuta cerca de la aplicación a través de localhost, y se recomienda que se ejecute bajo el mismo límite de red que la aplicación. Si bien muchos sistemas nativos de la nube hoy en día consideran el nivel de pod (en Kubernetes, por ejemplo) como un límite de seguridad confiable, Dapr proporciona al usuario autenticación de nivel de API mediante tokens.

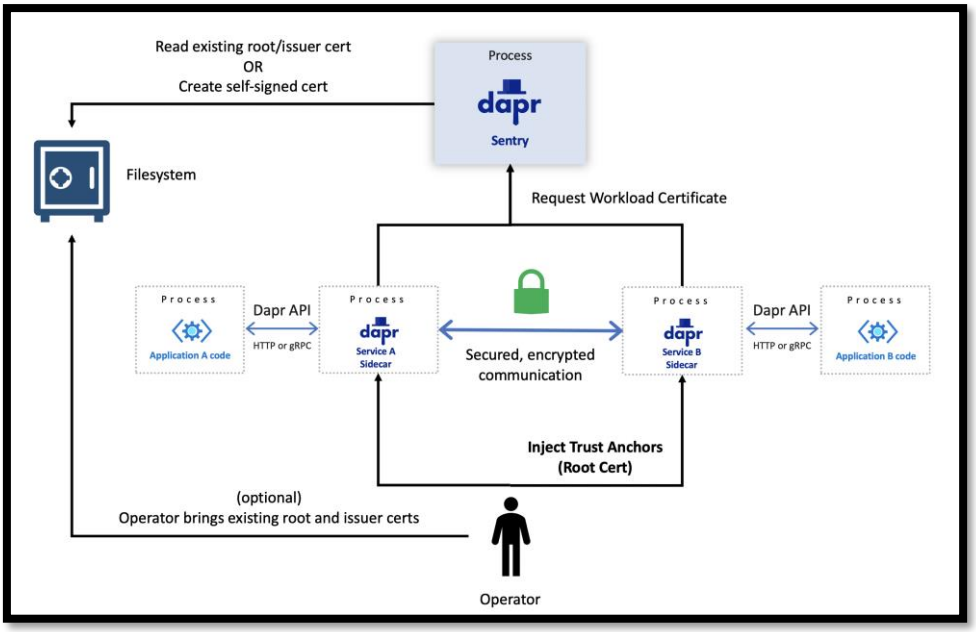
Comunicación de sidecar a sidecar

Dapr incluye un TLS mutuo automático "activado por defecto" que proporciona cifrado en tránsito para el tráfico entre los sidecars de Dapr. Para lograr esto, Dapr aprovecha un servicio de sistema denominado Sentryque actúa como una autoridad de certificación (CA) y firma las solicitudes de certificado de carga de trabajo (aplicación) que se originan en el sidecar de Dapr (Microsoft, 2019).

mTLS autohospedado

El diagrama de la figura 5 que se presenta a continuación, muestra cómo el servicio del sistema Sentry emite certificados para aplicaciones basados en el certificado raíz / emisor proporcionado por un operador o generado por el servicio Sentry almacenado en un archivo.

Figura 5 Diagrama de Sistema sentry almacenado en archivos.

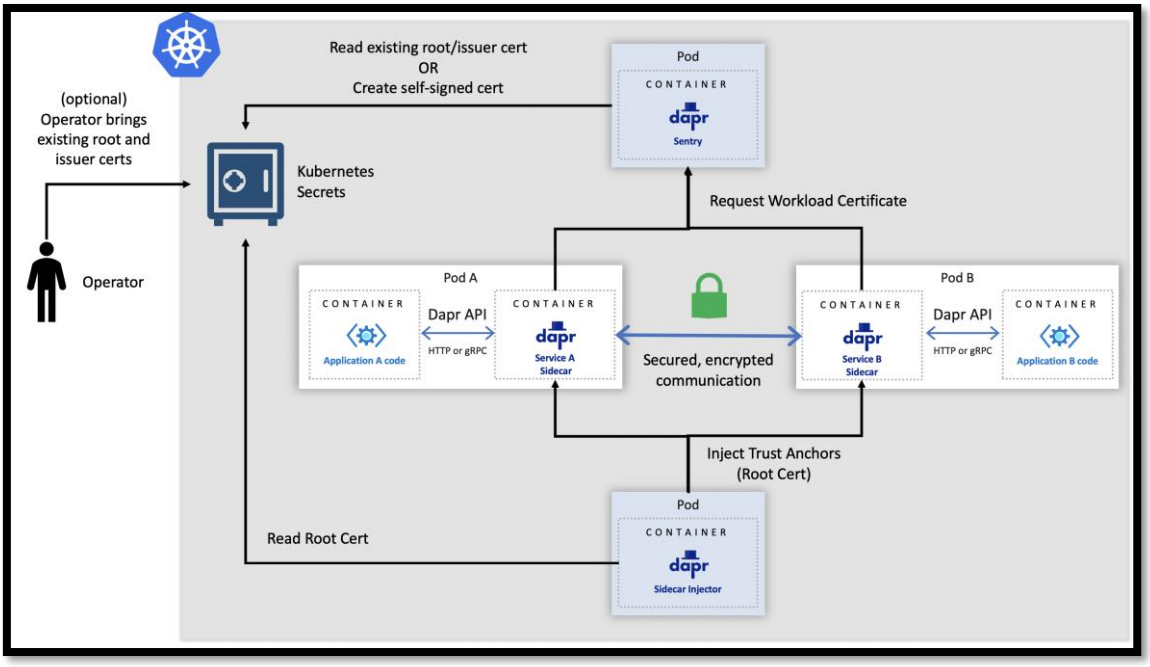


FUENTE: Dapr Docs. 2021. Overview. [online] Available at: <https://docs.dapr.io/concepts/overview/> [Accessed 22 March 2021].

mTLS en Kubernetes

El siguiente diagrama muestra cómo el servicio del sistema Sentry emite certificados para aplicaciones basados en el certificado raíz / emisor que proporciona un operador o generado por el servicio Sentry y se almacena como un secreto de Kubernetes.

Figura 6 Diagrama de Sistema sentry almacenado en secreto de Kubernetes



FUENTE: Dapr Docs. 2021. Overview. [online] Available at: <https://docs.dapr.io/concepts/overview/> [Accessed 22 March 2021].

Ámbitos y secretos del espacio de nombres de componentes

Los componentes de Dapr tienen un espacio de nombres. Eso significa que una instancia de sidecar en tiempo de ejecución de Dapr solo puede acceder a los componentes que se han implementado en el mismo espacio de nombres.

Los componentes de Dapr utilizan la capacidad de administración de secretos incorporada de Dapr para administrar los secretos.

Además, Dapr ofrece alcance a nivel de aplicación para componentes al permitir a los usuarios especificar qué aplicaciones pueden consumir componentes dados.

4.5 Metas fijadas con Dapr

Microsoft (2019), define los siguientes objetivos a alcanzar con el lanzamiento de Dapr como recurso open source:

- *Permitir a los desarrolladores que usen cualquier lenguaje o marco para escribir aplicaciones distribuidas*
- *Resolver los problemas difíciles que enfrentan los desarrolladores al crear aplicaciones de microservicio proporcionando bloques de construcción de mejores prácticas*
- *Impulsar el apoyo de la comunidad, abierto y neutral con respecto a los proveedores*
- *Obtener nuevos colaboradores*
- *Proporcionar consistencia y portabilidad a través de API abiertas*
- *Ser independiente de la plataforma en la nube y el perímetro*
- *Adoptar la extensibilidad y proporcione componentes enchufables sin la dependencia del proveedor*
- *Habilitar los escenarios de IoT y de borde al ser livianos y de alto rendimiento*
- *Ser adoptable de forma incremental a partir del código existente, sin dependencia del tiempo de ejecución.*[2]

4.6 Características

- *Sistema Pub-Sub impulsado por eventos con proveedores conectables y semántica al menos una vez*
- *Enlaces de entrada y salida con proveedores conectables*
- *Gestión de estado con almacenes de datos conectables*
- *Descubrimiento e invocación coherentes de servicio a servicio*
- *Modelos con estado opt-in: Consistencia fuerte / eventual, victorias de primera escritura / última escritura*
- *Actores virtuales multiplataforma*
- *Gestión de secretos para recuperar secretos de bóvedas de claves seguras*
- *Built-in Observabilidad apoyo*
- *Se ejecuta de forma nativa en Kubernetes con un operador dedicado y CRD*
- *Admite todos los lenguajes de programación a través de HTTP y gRPC*
- *Multinube, componentes abiertos (enlaces, pub-sub, estado) de Azure, AWS, GCP*
- *Se ejecuta en cualquier lugar, como proceso o en contenedores.*
- *Ligero (58 MB binarios, 4 MB de memoria física)*
- *CLI dedicada: experiencia amigable para el desarrollador con fácil depuración*
- *Clientes para Java, .NET Core, Go, Javascript, Python, Rust y C ++.*

[2] Channel 9. 2021. *Distributed Application Runtime (Dapr)*. [online] Available at: <https://channel9.msdn.com/Events/Build/2020/INT118> [Accessed 23 March 2021].

4.7 ANTECEDENTES

A continuación, se presenta el estado del arte en el que se encuentra actualmente esta plataforma que lleva en el mercado alrededor de 15 meses, sus resultados aún están en fase de implementación por parte de los desarrolladores del mundo, por lo cual, se presentan algunos micro proyectos que se han desarrollados aplicando Dapr y que han servido como proceso de validación para este nuevo producto de Open Source que Microsoft desea incorporar y en sus soluciones gratuitas para toda su comunidad digital, con la firme intención de demostrar las capacidades de Dapr utilizando diferentes lenguajes y abordar una amplia gama de escenarios comunes. Algunos se centran en patrones de uso específicos o en una capacidad de Dapr particular, mientras que otros son demostraciones de extremo a extremo que aprovechan varios componentes y capacidades de Dapr.

4.7.1 Proyecto de procesamiento de archivos por lotes

Esta componente permite observar una muestra de un extremo a otro para procesar un lote de archivos de texto relacionados mediante microservicios y Dapr. A través de ella, se pretende conocer todo lo relacionado con la administración de estados, las vinculaciones, Pub / Sub y el seguimiento de un extremo a otro de Dapr. Dicha muestra fue validada por última vez en Dapr v0.10.

Guión

La solución planteada por Schneider (2021), se presenta como respuesta a la necesidad de carga masiva de datos de una empresa de distribución, la compañía recibe órdenes de venta de distribuidores a través de archivos CSV, similar a la de muchas empresas. Estos archivos vienen en lotes. Cada lote se compone de exactamente tres archivos diferentes, cada uno con información diferente sobre los pedidos. Los archivos del mismo lote tienen el mismo prefijo, un formato de fecha y hora. Estos archivos pueden llegar en cualquier orden y en diferentes momentos, por lo que la empresa espera hasta que lleguen todos los archivos del mismo lote antes de procesarlos. El procesamiento de un lote consiste en combinar el contenido de estos tres archivos, convertirlos de CSV a un JSON por pedido y almacenar cada pedido en un almacén de datos.

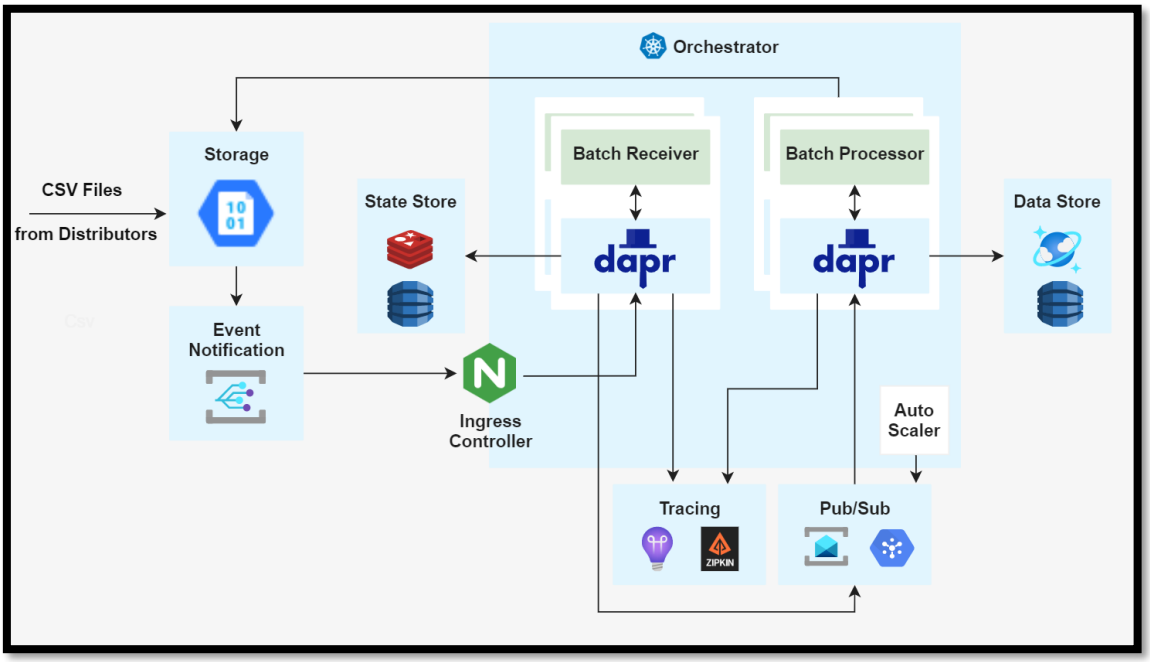
Solución

Tal como lo menciona su autor Schneider (2021), en este desarrollo, la implementación de muestra para el escenario de procesamiento de archivos por lotes combina Dapr y Azure. Como Dapr es independiente de la nube, sería posible volver a implementar la solución utilizando componentes distintos de Azure o que no sean de Azure. En este caso, se utilizan servicios de Azure como Service Bus y Cosmos DB, App Insight como backend para el seguimiento, pero se podrían utilizar

otros servicios; por ejemplo, en lugar de utilizar Service Bus como agente de mensajes, se podrían utilizar Redis Streams o Google Cloud Pub / Sub.

Como lo muestra Schneider (2021) en la figura 1, Azure Storage recibe lotes de archivos (del simulador Batch Generator). Cada archivo activará una notificación de Event Grid a un microservicio de Batch Receiver que utiliza la administración de estado para identificar cuándo han llegado todos los archivos de un lote. Luego, Pub / Sub se usa para activar un procesador por lotes que transformará los lotes en pedidos individuales basados en JSON y los guardará en Cosmos DB. El seguimiento de un extremo a otro de Dapr se utiliza para enviar información de seguimiento a App Insights. Pub / Sub se utiliza para proporcionar una capa de nivelación de carga y para escalar el procesador por lotes mediante KEDA, ya que el procesador realiza la mayor parte del trabajo de procesamiento.

Figura 7. Arquitectura de la solución de procesamiento por lotes



FUENTE: GitHub. 2021. dapr/samples. [online] Available at: https://github.com/dapr/samples/blob/master/batch-file-processing/images/solution-diagram-generic.png> [Accessed 22 March 2021].

Según la arquitectura propuesta por Microsoft (2021), tal como se observa en la figura 2, en esta implementación de referencia se utilizan los siguientes componentes.

Componentes

- *Storage- Azure Blob Storage es donde llegarán los lotes de archivos. Esta muestra incluye un generador de lotes para simular que los distribuidores envían lotes de archivos al almacenamiento.*

- *Event Grid - Event Grid se usa para enviar notificaciones de eventos de nuevos blobs (cada archivo en los lotes) al controlador de entrada de Kubernetes (NGINX en esta muestra).*

- *Kubernetes/Azure Kubernetes Service (AKS) - utilizado para organizar y alojar los contenedores y sus sidecars Dapr para los microservicios implementados por esta muestra, así como NGINX y la instancia de Redis utilizada por la solución.*
- *NGINX - el controlador de ingreso para Kubernetes usado en esta muestra que recibirá las notificaciones de Event Grid sobre nuevos blobs en almacenamiento y los enrutará al microservicio Batch Receiver.*

- *Batch Generator- un microservicio de aplicación de nodo que simula distribuidores que generan archivos CSV con información de pedidos y los envía Storagecada 1 min.*

- *Batch Receiver- un microservicio de aplicación de nodo que se ejecuta en Kubernetes y que se llama para cada archivo CSV. Tiene un sidecar Dapr para proporcionar funciones de seguimiento, administración de estado y Pub / Sub. Extrae el prefijo del archivo como una identificación de lote, luego usa esa identificación de lote como la clave para escribir en la administración de estado de Dapr, haciendo un seguimiento de los archivos que llegaron para cada lote. Utiliza la gestión estatal de Dapr Redispara obtener y actualizar el estado . Una vez que un lote tiene los 3 archivos, Batch Receiver colocará un mensaje en Dapr Pub / Sub (tema batchReceived) Service Buscomo intermediario de mensajes.*

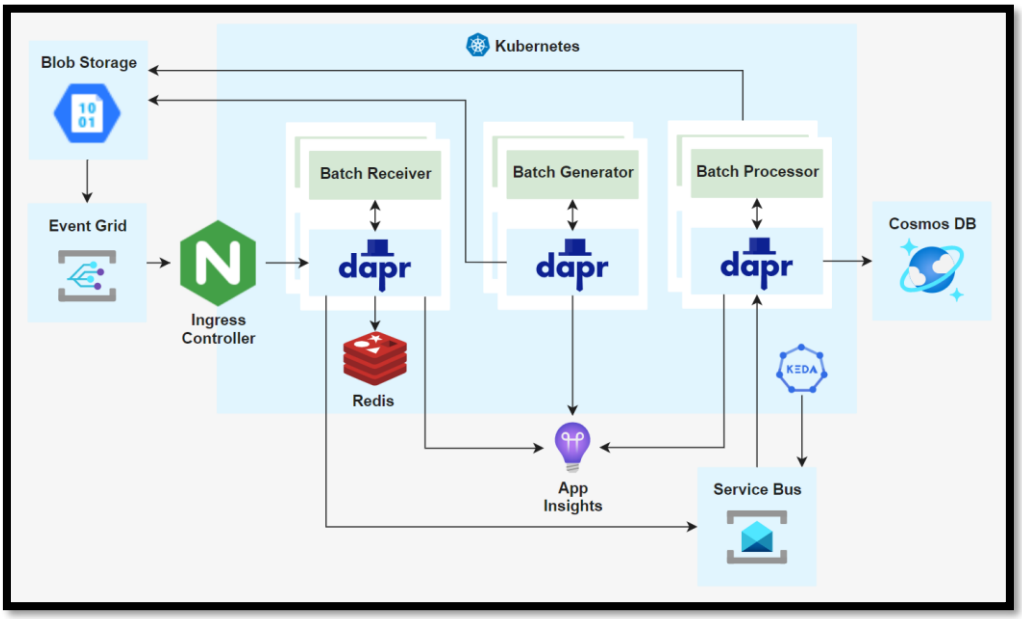
- *Batch Processor- un microservicio de aplicación de nodo activado desde Pub / Sub de Dapr. Tiene un sidecar Dapr para proporcionar funciones de seguimiento, Pub / Sub y enlace de salida. Se suscribe a los mensajes del tema batchReceived y cuando se recibe un mensaje, obtendrá los tres archivos para un lote del almacenamiento, los transformará en una matriz de pedidos JSON y almacenará cada pedido JSON enCosmos DB . Utiliza el enlace de salida Cosmos DB de Dapr para almacenar los datos. Este microservicio se escala usando KEDA.*

- *KEDA- Ajuste de escala automático impulsado por eventos basado en Kubernetes . Se usa para escalar horizontalmente el Procesador por lotes según la cantidad de mensajes sin procesar en el tema de Service Bus que se usa entre los microservicios para Pub / Sub.*

- *Cosmos DB- el almacén de datos NoSQL utilizado por la muestra para almacenar los pedidos individuales en formato JSON. Esto luego sería utilizado en sentido descendente por otras soluciones de la empresa.*

- *App Insights - Azure App Insights se usa para todo el seguimiento de un extremo a otro creado por Dapr para los microservicios de la solución.* [3]

Figure 8 - Componentes de la aplicación



FUENTE: Schneider, Y., 2021. dapr/samples. [online] GitHub. Available at: <https://github.com/dapr/samples/tree/master/batch-file-processing> [Accessed 22 March 2021].

4.7.2 Demostración de integración de administración de API de Dapr y Azure

En este desarrollo y tal como lo describe Microsoft (2021), la función de puerta de enlace autohospedada amplía el soporte de API Management para entornos híbridos y de múltiples nubes y permite a las organizaciones administrar de forma eficiente y segura las API alojadas en las instalaciones y en las nubes desde un único servicio de API Management en Azure.

Su implementación está soportada en el concepto general de puerta de enlace autohospedada, donde los clientes tienen la flexibilidad de implementar una versión en contenedor del componente de la puerta de enlace de administración de API en los mismos entornos donde alojan sus API.

[3] MICROSOFT, Componentes de la solución, 2017. Online.

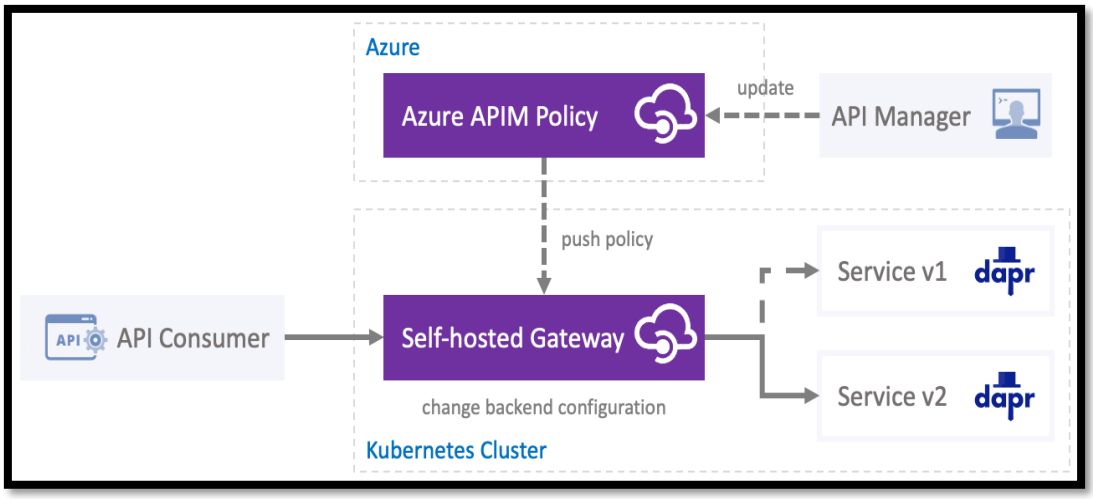
Todas las puertas de enlace autohospedadas se administran desde el servicio de administración de API con el que están federadas, lo que brinda a los clientes la visibilidad y la experiencia de administración unificada en todas las API internas y externas. Colocar las puertas de enlace cerca de las API permite a los clientes optimizar los flujos de tráfico de las API y abordar los requisitos de seguridad y cumplimiento.

En esta demostración, se pretende analizar la configuración del servicio API Management y su puerta de enlace autohospedada en Kubernetes. Para ilustrar la integración de Dapr, revisando así, tres casos de uso de Dapr:

1. Invocación de un método de servicio Dapr específico
2. Publicar contenido en un tema de Pub / Sub
3. Vinculación de invocación con transformación de contenido de solicitud

Como se observa en la figura 8, la puerta de enlace recibe todas las solicitudes y normalmente las reenvía sin modificar a la API subyacente. Sin embargo, una directiva puede aplicar cambios a la solicitud de entrada y a la respuesta de salida.

Figura 8 - Gestión de políticas



FUENTE: Docs.microsoft.com. 2021. *Azure API Management*. [online] Available at: <https://docs.microsoft.com/en-us/rest/api/apimanagement/> [Accessed 22 March 2021].

En resumen, esta demostración fue diseñada con el objetivo de comprender la forma de configurar el servicio APIM e implementar la puerta de enlace autohospedada en su clúster para administrar el acceso a cualquier número de servicios Dapr alojados en Kubernetes.

4.7.3 Dapr con Docker-Compose

Esta solución fue diseñada con el fin de demostrar cómo Dapr permite la ejecución localmente con Docker Compose, que es una herramienta para definir y ejecutar aplicaciones Docker de varios contenedores.

Como lo indica Microsoft (2017), Compose funciona en todos los entornos: producción, puesta en escena, desarrollo, pruebas y flujos de trabajo de CI y posee comandos para administrar todo el ciclo de vida de su aplicación:

- Iniciar, detener y reconstruir servicios
- Ver el estado de los servicios en ejecución
- Transmita la salida del registro de los servicios en ejecución
- Ejecute un comando único en un servicio

Este desarrollo tiene un gran valor agregado gracias a que ofrece de las principales características que ofrece Compose, se enumeran:

Múltiples entornos aislados en un solo host: permite múltiples entornos aislados en un solo host:

- Host de desarrollo, para crear varias copias de un solo entorno, como cuando se desea ejecutar una copia estable para cada rama de función de un proyecto.

- En un servidor de CI, para evitar que las compilaciones interfieran entre sí, puede establecer el nombre del proyecto en un número de compilación único.

- En un host compartido o dev host, para evitar que diferentes proyectos, que pueden usar los mismos nombres de servicio, interfieran entre sí.

Conservar los datos de volumen cuando se crean contenedores: Compose conserva todos los volúmenes utilizados por sus servicios. Cuando se docker-compose up ejecuta, si encuentra contenedores de ejecuciones anteriores, copia los volúmenes del contenedor antiguo al contenedor nuevo. Este proceso garantiza que los datos que haya creado en volúmenes no se pierdan. (Microsoft, 2017).

Variables y movimiento de una composición entre entornos: Compose admite variables en el archivo Compose. Puede utilizar estas variables para personalizar su composición para diferentes entornos o diferentes usuarios.

Casos de uso comunes: Compose se puede utilizar de muchas formas diferentes. Algunos casos de uso comunes se describen a continuación.

- **Entornos de desarrollo:** Cuando está desarrollando software, la capacidad de ejecutar una aplicación en un entorno aislado e interactuar con ella es crucial. La herramienta de línea de comandos Redactar se puede utilizar para crear el entorno e interactuar con él.

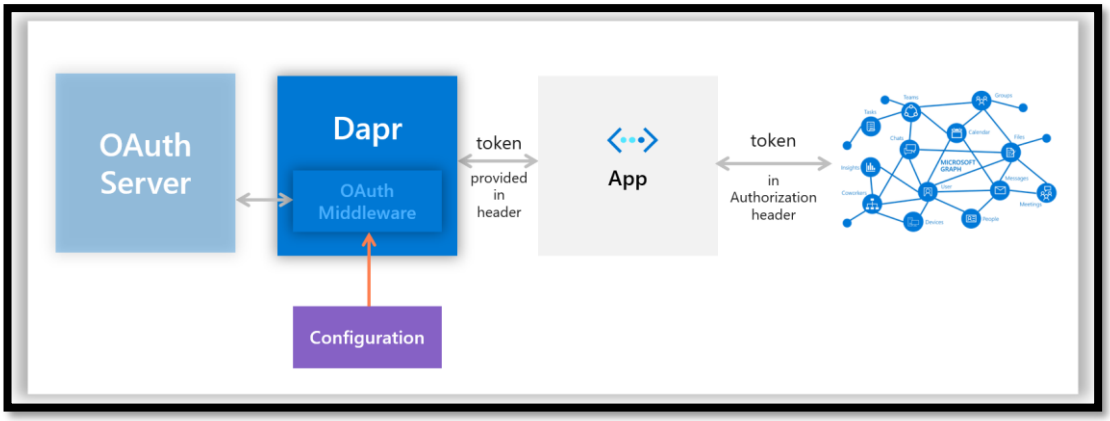
  El archivo Compose proporciona una forma de documentar y configurar todas las dependencias de servicio de la aplicación. Con la herramienta de línea de comandos Redactar, puede crear e iniciar uno o más contenedores para cada dependencia con un solo comando (docker-compose up). Juntas, estas características brindan una manera conveniente para que los desarrolladores comiencen a trabajar en un proyecto.

- **Entornos de prueba automatizados:** Una parte importante de cualquier proceso de implementación o integración continua es el conjunto de pruebas automatizado. Las pruebas automatizadas de un extremo a otro requieren un entorno en el que ejecutar las pruebas. Compose proporciona una forma conveniente de crear y destruir entornos de prueba aislados para su suite de pruebas.

4.7.4 Autorización OAuth para servicio externo (no interactivo)

Este desarrollo, está basado en la configuración el middleware OAuth para permitir que un servicio interactúe con servicios externos que requieren autenticación.
Como se puede visualizar en la figura 9, este diseño separa las preocupaciones de autenticación / autorización de la aplicación.

Figura 9 Diagrama de arquitectura Dapr



FUENTE: GitHub. 2021. *dapr/samples*. [online] Available at: <https://github.com/dapr/samples/blob/master/middleware-clientcredentials/img/architecture_diagram.png> [Accessed 22 March 2021].

Microsoft Azure (2017), ha demostrado como en los últimos años y con el auge del uso de las APIs y el interés por parte de las empresas en incorporar plataformas API Management dentro del concepto del API Economy en sus organizaciones, surge la necesidad de securizar las APIs, es por eso que surge la estrategia de securización de APIs debe encontrar el equilibrio entre la seguridad y la usabilidad.

El uso de estándares ayuda a conseguir ese equilibrio, ya que son suficientemente complejos como para disuadir a usuarios maliciosos y, proporcionando la información correcta, permiten el acceso a usuarios autorizados.

La empresa Oauth.net (2021) nos habla acerca de las definiciones que se deben considerar al momento de enviar paquetes de datos mediante una red de aplicaciones y API habilitadas para la web, entre sus consideraciones se tienen las siguientes:

> ***Autenticación vs autorización***
>
> *La especificación OAuth 2.0 define un protocolo de delegación que es útil para transmitir decisiones de autorización a través de una red de aplicaciones y API habilitadas para la web. OAuth se utiliza en una amplia variedad de aplicaciones, incluido el suministro de mecanismos para la autenticación de usuarios. Esto ha llevado a muchos desarrolladores y proveedores de API a concluir incorrectamente que OAuth es en sí mismo un protocolo de autenticación y a utilizarlo erróneamente como tal.*
>
> *Con frecuencia, cuando se habla de seguridad y de OAuth, los conceptos de autenticación y autorización se solapan y son completamente diferentes:*
> - *La autenticación se define como el proceso mediante el cual se verifica quién eres, es decir, su ámbito se refiere a la identificación.*
> - *La autorización es el proceso mediante el cual se verifica a qué tienes acceso, es decir, su ámbito se limita al control de acceso.*[4]

De esta manera, se evidencia como mediante el uso de Dapr como parte de la composición de arquitectura para proyectos, se potencia en gran medida la seguridad de las APIs y se garantiza que el proceso de autenticación y autorización también puede ser administrado de manera simple y eficaz dentro de cada uno de los desarrollos.

4.7.5 Leer eventos de Kubernetes

Este desarrollo se plantea en enero del año 2021, adaptando las condiciones de ejecución de Dapr con un enlace de entrada de eventos de Kubernetes, está desarrollada en lenguaje Node y cuenta además con una definición de componente de enlace de eventos de Kubernetes.

Para comprender mejor los beneficios de este proyecto, se deberán reconocer conceptos técnicos de desarrollo, que Nogués García (2018), nos presenta a continuación:

> ***Kubernetes:*** *es una plataforma portable y extensible de código abierto para administrar cargas de trabajo y servicios. Kubernetes facilita la automatización y la configuración declarativa. Tiene un ecosistema grande y en rápido crecimiento. El soporte, las herramientas y los servicios para Kubernetes están ampliamente disponibles.*
>
> *Kubernetes tiene varias características, entre ellas se encuentran:*
>
> - *Una plataforma de contenedores*
> - *Una plataforma de microservicios*
> - *Una plataforma portable de nube*
>
> *Esta plataforma ofrece un entorno de administración centrado en contenedores, de forma tal que orquesta la infraestructura de cómputo, redes y almacenamiento para que las cargas de trabajo de los usuarios no tengan que hacerlo, llegando a ofrecer la simplicidad de las Plataformas como Servicio (PaaS) con la flexibilidad de la Infraestructura como Servicio (IaaS) y permite la portabilidad entre proveedores de infraestructura.*[5]

4.8 Benchmark de rendimiento

Rendimiento de invocación de servicio

En este apartado se proporcionan las evaluaciones comparativas de rendimiento de la API de invocación de servicios y utilización de recursos para los componentes necesarios para ejecutar Dapr en diferentes entornos de alojamiento.

Resumen del sistema

Dapr consta de un plano de datos, el sidecar que se ejecuta junto a su aplicación y un plano de control que configura los sidecars y proporciona capacidades como la gestión de identidades y certificaciones.

Componentes autohospedados

- Sidecar (plano de datos)
- Centinela (opcional, avión de control)
- Colocación (opcional, plano de control)

[4] OAuth 2.0, Implementar el código de autorización OAuth 2.0. Online.

[5] Nogués García, J., 2021. Orquestación de contenedores con Kubernetes. [online] E-archivo.uc3m.es. Available at: <https://e-archivo.uc3m.es/handle/10016/29528#preview> [Accessed 22 March 2021].

Componentes de Kubernetes

- Sidecar (plano de datos)
- Centinela (opcional, avión de control)
- Colocación (opcional, plano de control)
- Operador (plano de control)
- Inyector Sidecar (plano de control)

Resumen de rendimiento para Dapr v1.0

La API de invocación de servicios es un proxy inverso con descubrimiento de servicios integrado para conectarse a otros servicios. Esto incluye rastreo, métricas, mTLS para el cifrado en tránsito del tráfico, junto con la resistencia en forma de reintentos para particiones de red y errores de conexión.

Con la invocación de servicio, puede llamar de HTTP a HTTP, de HTTP a gRPC, de gRPC a HTTP y de gRPC a gRPC. Dapr no usa HTTP para la comunicación entre sidecars, siempre usa gRPC, mientras que transfiere la semántica del protocolo usado cuando se llama desde la aplicación. La invocación del servicio es el mecanismo subyacente de comunicación con los actores de Dapr.

Configuración de la prueba de rendimiento de Kubernetes

La prueba se realizó en un clúster de Kubernetes de 3 nodos, utilizando hardware básico con 4 núcleos y 8 GB de RAM, sin ninguna aceleración de red.
La configuración incluía un pod de prueba de carga (Fortio) con un sidecar Dapr inyectado en él que llamaba a la API de invocación del servicio para llegar a un pod en un nodo diferente.

Parámetros de prueba:

1000 solicitudes por segundo
Sidecar limitado a 0,5 vCPU
Sidecar mTLS habilitado
Telemetría de sidecar habilitada (seguimiento con una frecuencia de muestreo de 0,1)
Carga útil de 1 KB

La prueba de referencia incluyó tráfico directo, no cifrado, sin telemetría, directamente desde el probador de carga a la aplicación de destino.

Rendimiento del plano de control

El plano de control de Dapr utiliza un total de 0,009 vCPU y 61,6 Mb cuando se ejecuta en modo no HA, lo que significa una única réplica por componente del sistema. Cuando se ejecuta en una configuración de producción de alta disponibilidad, el plano de control de Dapr consume ~ 0.02 vCPU y 185 Mb.

Tabla 1 Rendimiento del plano de control

Componente	CPU virtual	Memoria
Operador	0,001	12,5 Mb
Centinela	0,005	13,6 Mb
Inyector Sidecar	0,002	14,6 Mb
Colocación	0,001	20,9 Mb

Hay una serie de variantes que afectan la CPU y el consumo de memoria para cada uno de los componentes del sistema. Estas variantes se muestran en la tabla 2.

Tabla 2 Componentes del sistema

Componente	CPU virtual	Memoria
Operador	Cantidad de pods que solicitan componentes, configuraciones y suscripciones	
Centinela	Número de solicitudes de certificado	
Inyector Sidecar	Número de solicitudes de admisión	
Colocación	Número de operaciones de reequilibrio de actores	Número de hosts de actores conectados

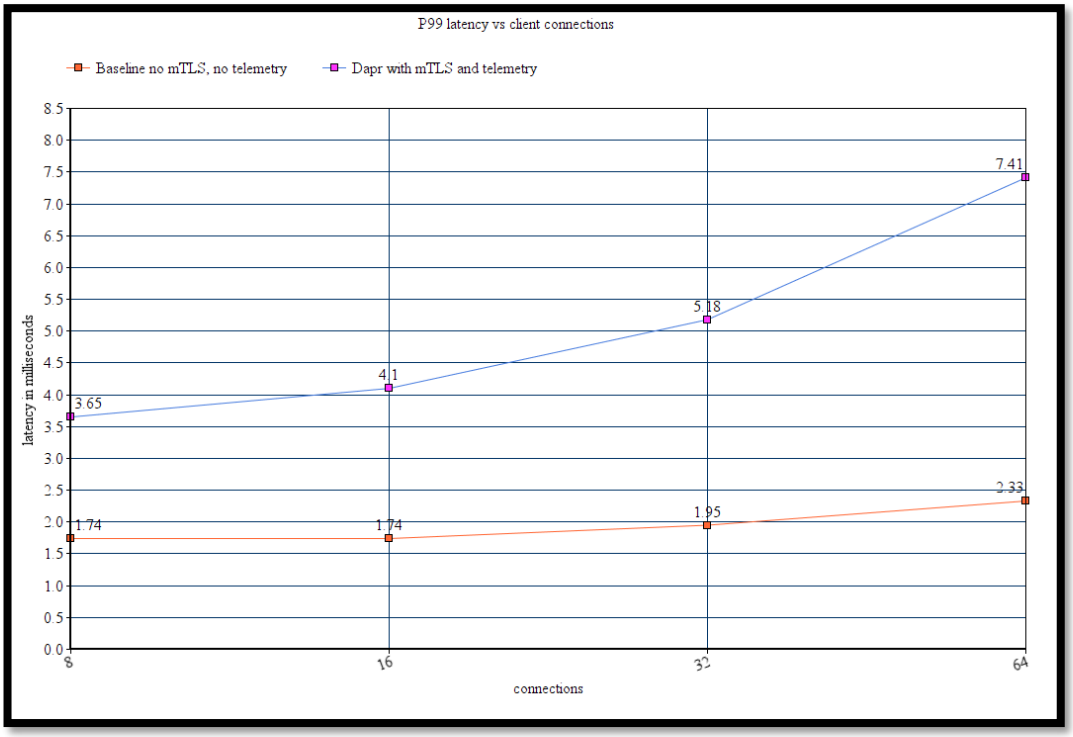
Rendimiento del plano de datos

El sidecar de Dapr usa 0.48 vCPU y 23Mb por cada 1000 solicitudes por segundo. De un extremo a otro, los sidecars de Dapr (cliente y servidor) agregan ~ 1,40 ms a la latencia del percentil 90 y ~ 2,10 ms a la latencia del percentil 99. Aquí hay una llamada de un extremo a otro de una aplicación a otra que recibe una respuesta. Esto se muestra en los pasos 1-7 de este diagrama. Este rendimiento está a la par o mejor que las mallas de servicio de uso común.

Latencia

En la configuración de prueba, las solicitudes pasaron por el sidecar de Dapr tanto en el lado del cliente (que atiende solicitudes de la herramienta de prueba de carga) como en el lado del servidor (la aplicación de destino). Se habilitaron mTLS y telemetría (rastreo con una frecuencia de muestreo de 0.1) y métricas en la prueba Dapr y se deshabilitaron para la prueba de línea de base.

Figura 10 Latencia en milisegundos por conexiones



5. METODOLOGÍA

5.1 Tipo de trabajo

Este proyecto se desarrolla mediante investigación aplicada que identifica un problema a intervenir y desarrolla posibles alternativas de solución, en este caso se identificó como problema la dificultad que representa para los desarrolladores de la fábrica de software de la facultad de ingeniería de Sistemas y Telecomunicaciones de la Universidad de Manizales, la integración de diversos lenguajes de programación mediante el uso de una malla de microservicios para manejar el alto tráfico en las aplicaciones.

El límite objetivo tiene un alcance especifico de desarrollo de una aplicación de baja complejidad codificada bajo los estándares de dos lenguajes de programación altamente reconocidos en el mundo de la ingeniería, siendo estos. NetCore y NodeJs y que use una malla de microservicios para su gestión, permitiendo así el despliegue de Dapr como entorno de ejecución en tiempo real para comunicación entre ambas aplicaciones.

5.2 Procedimiento

El proyecto se llevará a cabo en cuatro fases, que permitirán el desarrollo y correcta ejecución de los pasos requeridos para comprender, analizar e implementar de forma coherente el objetivo propuesto.

5.2.1 Fase 1. PLANEACIÓN. mediante reuniones constantes se programaron las actividades que hacen parte del entregable final y que apuntan a la consecución de los objetivos planteados.

- Actividad 1. **Definición de alcance:** identificar los lenguajes que se quieren contemplar en el desarrollo y el nivel de profundidad de la investigación acerca de la herramienta Dapr, que para el caso son Python y NodeJs.

- Actividad 2. **Definición de objetivos:** identificación de cada uno de los objetivos que esperan cumplir, teniendo en cuenta las limitaciones del estado del arte y su existencia en el mercado.

- Actividad 3. **Definición de recursos:** se requiere de una evaluación del contenido de Dapr para establecer la viabilidad de utilizar determinados recursos o lenguajes de programación en la implementación.

5.2.2 Fase 2. ANÁLISIS Y DISEÑO: se hace necesario identificar dentro de todas las posibilidades que ofrece Dapr, cual arquitectura de desarrollo se desea implementar, enfocando las necesidades a la integración y uso de una malla de microservicios que facilite el consumo de las apis desarrolladas e implementadas.

- Actividad 1. **Definición Arquitectura:** se implementa mediante el uso de microservicios, de forma tal que sean aplicaciones más adaptables y flexibles. Porque si se quiere modificar solamente un servicio, no es necesario alterar el resto de la infraestructura. Cada uno de los servicios se puede desplegar y modificar sin que ello afecte a otros servicios o aspectos funcionales de la aplicación.

- Actividad 2. **Capacitación en tecnologías requeridas:** al ser Dapr una tecnología relativamente nueva, se hace necesario indagar mucho en el estado del arte para encontrar documentación y manuales de instalación y uso de la misma, por lo que la capacitación se vuelve un tema extenso y requiere de varios días para su finalización.

5.2.3 Fase 3. DESARROLLO E IMPLEMENTACIÓN:

- Actividad 1. **Configuración de entornos de trabajo:** El ambiente de trabajo en el que se desarrollan las funciones impacta en los desarrollos de resultados que se obtiene, se configuran los IDE para desarrollo de Visual Studio 2019 y Sublime para NodeJs, cada uno de acuerdo a las preferencias y necesidades de políticas y las configuraciones que incluyen en el desempeño del equipo.

- Actividad 2. **Desarrollo API Python: Crear un proyecto de API web.** Se agregar las clase de modelos y un contexto de base de datos, se debe aplicar scaffolding a un controlador con métodos CRUD y finalmente se configura el enrutamiento, las rutas de acceso URL y los valores devueltos.

- Actividad 3. **Desarrollo API NodeJs:** se instala y configura npm y git, se procede a crear repositorio en GitHub y clonar el mismo, se diseña y crea la estructura de la aplicación y finalmente se testea la funcionalidad que se desea implementar.

- Actividad 4. **Integración APIS con Dapr:** este proceso por definición es uno de los más importantes, en el se se deben seguir los siguientes pasos:

    - Ejecución de una **instancia de contenedor de Redis** para utilizarla como tienda estatal local y agente de mensajes
    - Ejecución de una **instancia de contenedor Zipkin** para la observabilidad.

- Crear una **carpeta de componentes predeterminada** con definiciones de componentes para lo anterior
- Ejecución de una **instancia de contenedor del servicio de colocación de Dapr** para la compatibilidad con actores locales

Para llevar a cabo este proceso se deben cumplir los siguientes requisitos:

- Dapr CLI con Dapr inicializado
- Node.js versión 8 o superior y / o Python 3.4 o superior, puede ejecutar este inicio rápido con uno o ambos microservicios

5.2.3 Fase 4. DOCUMENTACIÓN: Diseñar y elaborar el documento final según los estándares establecidos.

- Actividad 1. **Elaboración de estructura documento.** Se construye con base al documento recibido por parte del programa de Ingeniería de sistemas y telecomunicaciones como opción parcial para grado, identificando cada una de las secciones y llevando a bien fin el objetivo planteado en el mismo.

- Actividad 2. **Documento técnico Dapr** se elaborará un documento final que contenga los apartados técnicos de implementación de APIs con microservicios y plataforma Dapr necesarios para su consecución y replica en el momento en que se requiera.

6. RESULTADOS

6.1 DESCRIPCIÓN DE RESULTADOS

Se considera como principal resultado obtenido la identificación de las principales ventajas del desarrollo de aplicaciones nativas en la nube mediante el uso de Dapr en arquitecturas con mallas de microservicios, que en la actualidad tiene mucha demanda y han surgido como respuesta a determinadas necesidades del mercado y que servirá como base para los futuros desarrollos de aplicaciones con condiciones semejantes por parte de los integrantes de la fábrica de software de la Universidad de Manizales.

De igual forma, el establecimiento de un benchmark que permitió medir e identificar claramente el rendimiento de invocación de servicios y utilización de recursos de la plataforma DAPR en diferentes entornos de alojamiento para conocer su capacidad en ejecución.

Para ello, se presentan los resultados intermedios y sus tiempos de ejecución:

Objetivo No.	Resultado obtenido	Medio de verificación	Semana de obtención
1.	Entorno de trabajo completo para desarrollo. NetCore y para desarrollo NodeJs	Implementación de método demo para evidenciar funcionamiento correcto.	Semana 2
2.	Entorno de trabajo completo para integración. NetCore y NodeJs con la plataforma Dapr	Implementación de método demo para evidenciar funcionamiento correcto de Dapr	Semana 4
3.	Implementación de arquitectura de microservicios en cada una de las APIs.	Validación de llamado de métodos API mediante la herramienta postman.	Semana 6
4.	Integración de APIs utilizando arquitectura de microservicios y plataforma Dpar	Ejecución en línea de peticiones http con respuestas automáticas del servidor.	Semana 8
5.	Inicio documentación ventajas y desventajas de implementar Dapr como entorno de ejecucion en tiempo real.	Documento con la estructura definida y la conceptualización pertinente.	Semana 10
6.	Continuación documentación ventajas y desventajas de implementar Dapr como	Documento con la estructura definida y la	Semana 12

	entorno de ejecución en tiempo real.	conceptualización pertinente.	
7.	Fin de documentación manual técnico para implementar Dapr como entorno de ejecución en tiempo real.	Documento con la estructura definida y la conceptualización pertinente.	Semana 14
8.	Benchmark del Benchmark de invocación de servicios y utilización de recursos de la plataforma DAPR	Benchmark construido y estructurado para validación de consumo de recursos con Dapr.	Semana 16

6.2 DISCUSIÓN DE RESULTADOS

Teniendo en cuenta los resultados obtenidos, es posible determinar la viabilidad de establecer proyectos de desarrollo de software con mallas de microservicios en poco tiempo y sin hacer distinción de lenguajes de programación, esta capacidad que ofrece Dapr, le brinda a los desarrolladores la posibilidad de ofrecer mayor rendimiento en sus procesos y ante todo en la obtención pronta de un producto mínimo viable.

Cuando se establece un benchmark de rendimiento en la invocación de servicios y la utilización de recursos de la plataforma Dapr, se pueden identificar los tiempos de ejecución mejorados en comparación con otras arquitecturas de microservicios, llegando a tener 1000 solicitudes por segundo y tan solo un consumo de en cada sidecar de Dapr de 0.48 vCPU y 23Mb de memoria.

7. CONCLUSIONES

- Al utilizar Dapr como tiempo de ejecución de desarrollo, se llega a conocer el verdadero potencial que tiene esta herramienta ayudando a optimizar recursos a la hora de gestionar los numerosos desafíos inherentes al desarrollo de aplicaciones de microservicios, que normalmente son impulsados por eventos, haciendo ejecuciones de código en respuesta a eventos como disparadores de telemetría, interacciones de usuario, entre otros componentes.

- La integración de aplicaciones desarrolladas en diferentes lenguajes de programación, ya no se traduce en un reto para sus creadores, quienes mediante la implementación de Dapr en sus proyectos, suplen la necesidad de acoplar diferentes ambientes para aplicaciones nativas que se produzcan en el ámbito de mallas de microservicios y que se construyan en diversos lenguajes y patrones de desarrollo.

- Dapr ofrece un completo paso a paso de implementación de buenas prácticas para implementar microservicios en bloques de construcción que son independientes, abiertos y exponen APIs para una fácil integración y una clara separación de componentes que se vuelven mucho más administrables para los actores del sistema.

- Los proyectos open source apoyados por compañías como Microsoft, brindan la posibilidad a todos los desarrolladores de generar conocimiento en grandes comunidades abiertas que facilitan el acceso a tecnologías de punta y que permiten robustecer conjuntamente cada componente mediante la interacción de actores que determinan la confiabilidad y la puesta a punto de cada elemento del mismo.

8. RECOMENDACIONES

- Se recomienda la incorporación de Dapr y la implementación de buenas prácticas de desarrollo en cada uno de los proyectos ejecutados dentro de la fábrica de software de la Universidad de Manizales, ya que contribuye a la optimización de recursos tanto de hardware como de software y brinda un estándar de calidad avalado por una de las más importantes empresas de tecnología y desarrollo a nivel mundial.

- Se recomienda realizar procesos de validación de recursos en máquina utilizados antes y después de la implementación de proyectos con Dapr, para conocer el impacto en latencia, variantes que afectan la CPU y el consumo de memoria para cada uno de los componentes del sistema, de forma tal que se puedan garantizar y apropiar correctamente dentro de cada uno de los proyectos que se pretendan llevar a cabo.

- Se recomienda la implementación de los proyectos Dapr con arquitectura de Kubernetes pues brinda escalabilidad y balanceo de carga, aislamiento de procesos y aplicaciones, facilidades para el despliegue y optimización de recursos automáticos por lo que se ven potenciadas en su máxima expresión las bondades de ambas tecnologías, además de contar con la posibilidad de soluciones en la nube tales como Google Cloud Platform.

BIBLIOGRAFÍA

Amazon Web Services. (2019). ¿Qué son los microservicios? | AWS. Amazon Web Services, Inc. https://aws.amazon.com/es/microservices/

Channel 9. 2021. Distributed Application Runtime (Dapr). [online] Available at: <https://channel9.msdn.com/Events/Build/2020/INT118> [Accessed 23 March 2021].

Docs.microsoft.com. 2021. Azure API Management. [online] Available at: <https://docs.microsoft.com/en-us/rest/api/apimanagement/> [Accessed 22 March 2021].

GitHub. 2021. dapr/samples. [online] Available at: <https://github.com/dapr/samples/tree/master/batch-file-processing> [Accessed 23 March 2021].

J. Lewis and M. Fowler, “Microservices,” 2014. [Online]. Available: https://martinfowler.com/articles/microservices.html. [Accessed: 13-Feb-2021].

López, D., & Maya, E. (2017). Arquitectura de Software basada en microservicios para desarrollo de aplicaciones Web.

Nebel, A. (2019.). Arquitectura de microservicios para plataformas de integración. Tesis de maestría. Universidad de la República (Uruguay). Facultad de Ingeniería.

Nogués García, J., 2021. Orquestación de contenedores con Kubernetes. [online] E-archivo.uc3m.es. Available at: <https://e-archivo.uc3m.es/handle/10016/29528#preview> [Accessed 22 March 2021].

Oauth.net. 2021. End User Authentication with OAuth 2.0 — OAuth. [online] Available at: <https://oauth.net/articles/authentication/> [Accessed 22 March 2021].

Pautasso, O. Zimmermann, M. Amundsen, J. Lewis, and N. Josuttis, “Microservices inPractice, Part 1: Reality Check and Service Design,” IEEE Softw., vol. 34, no. 1, pp. 91– 98, 2017

ANEXO A
DESARROLLO WEB CON DAPR

ANEXO A. DESARROLLO WEB CON DAPR

El presente documento contiene los diferentes componentes relacionados con el tema del proyecto y que permiten la comprensión y ejecución del mismo.

Figura 1. Arquitectura de la solución A

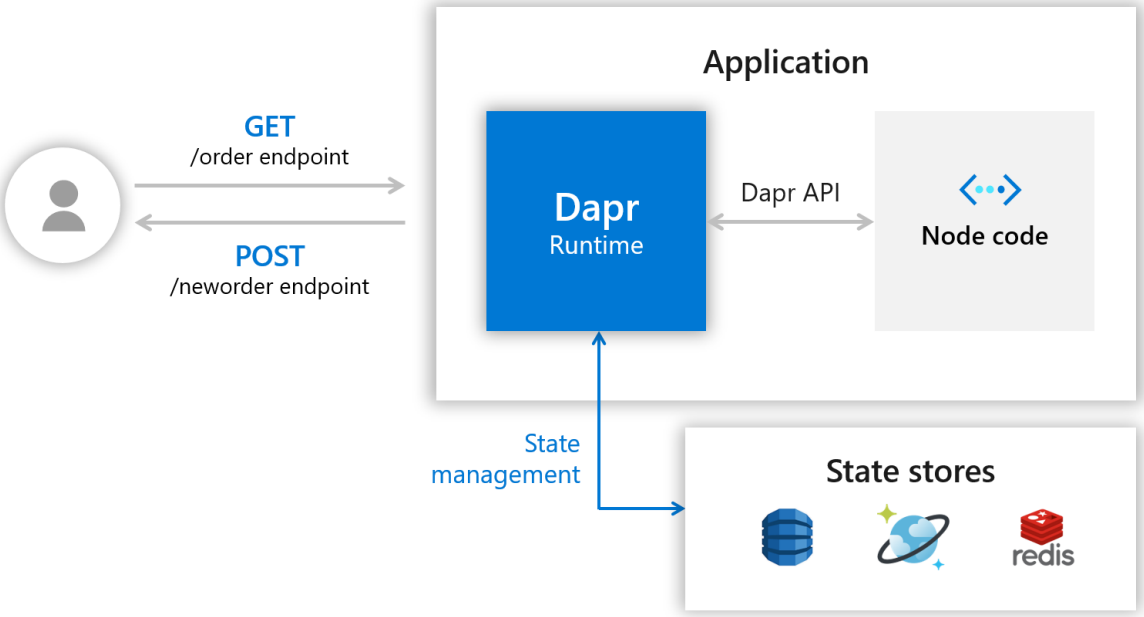
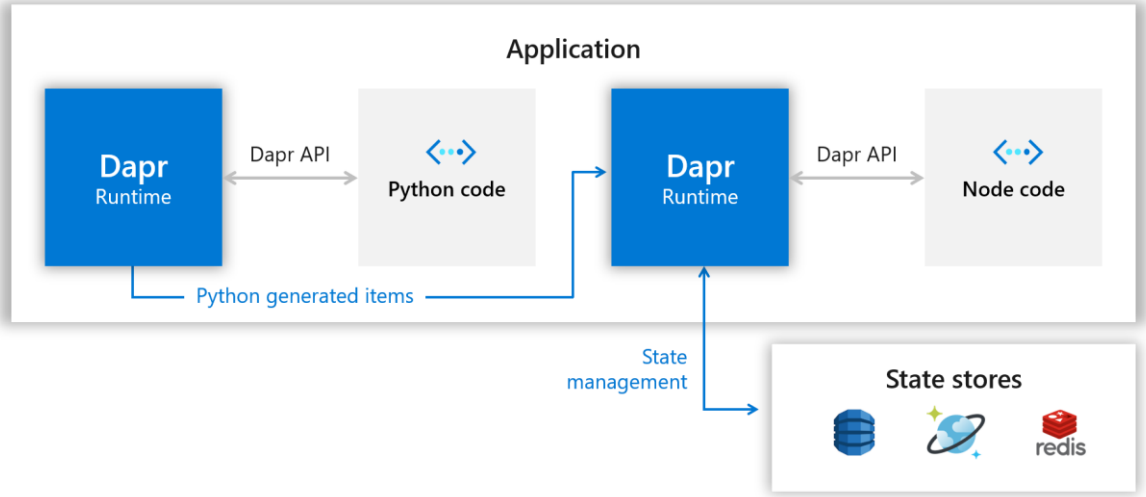


Figura 2. Diagrama de arquitectura B

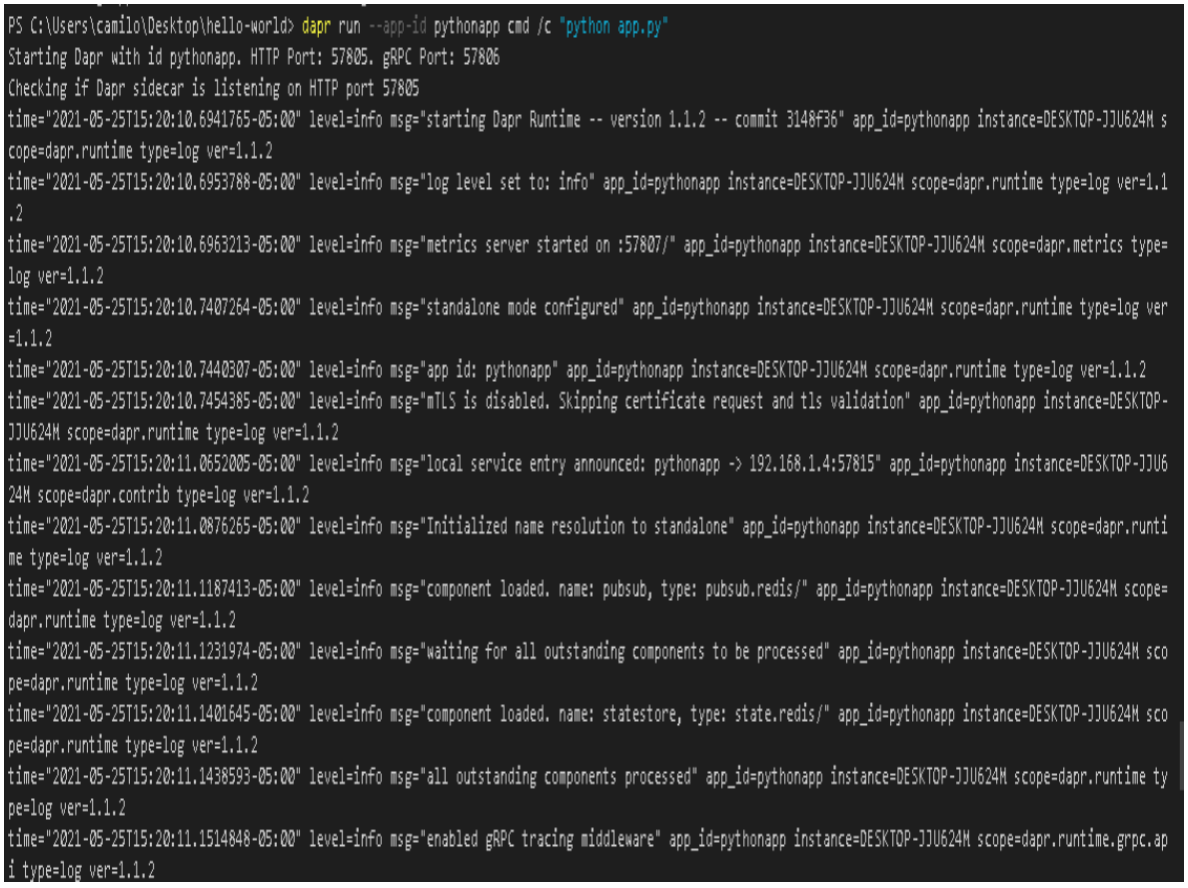


1. PYTHON

A continuación, se reflejan las respuestas a partir de las peticiones realizadas a la a aplicación desarrollada con Python para la invocación de servicios y la verificación de estado permanente en la integración con Dapr.

1.1 Ejecución en tiempo real con Dapr

Figura 3. Ejecución en tiempo real Python con Dapr



2. NODEJS

A continuación, se reflejan las respuestas a partir de las peticiones realizadas a la a aplicación desarrollada con NodeJs para la invocación de servicios y la verificación de estado permanente en la integración con Dapr.

1.2 Ejecución en tiempo real con Dapr

Figura 4. Ejecución en tiempo real NodeJs con Dapr