UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y EN SISTEMAS

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

Uso de tokens para la autorización de usuarios de HTCondor-CE en el entorno GRID UNAM

TESINA

QUE PARA OBTENER EL GRADO DE
ESPECIALISTA EN CÓMPUTO DE ALTO RENDIMIENTO

PRESENTA:
OSCAR ORTEGA BLANCAS

TUTOR-DIRECTOR DE TESINA:
DR. LUKAS NELLEN FILLA

IIMAS
UNAM

Ciudad Universitaria, Ciudad de México 6 junio, 2024

Agradecimientos

Expreso mi sincero agradecimiento a mis queridos padres, Donato Ortega y Crispina Blancas. Mi más profundo agradecimiento por su constante apoyo, paciencia y respaldo. Su amor, confianza y sacrificio han sido invaluables en mi camino académico y profesional.

De igual manera a los profesores Dr. Lukas Nellen Filla y MC. Juan Luciano Díaz González, y en especial al Lic. Pedro Damián Cruz Santiago, por su invaluable asesoría en la realización de este proyecto. Su dedicación, tiempo y comprensión han sido fundamentales para el éxito de este trabajo. Agradezco profundamente su guía, consejos y apoyo a lo largo de este proceso. Su experiencia y conocimientos han sido una invaluable contribución para el desarrollo de mi proyecto. ¡Muchas gracias por su inestimable colaboración!".

Me encuentro profundamente agradecido con la Universidad Nacional Autónoma de México por brindarme la valiosa oportunidad de estudiar y desarrollarme profesionalmente. A lo largo de mi trayectoria académica, la universidad me ha proporcionado las bases y los conocimientos fundamentales que han sido esenciales para mi crecimiento y desarrollo.
Además, quiero agradecer de manera especial a todos aquellos profesores que han compartido su valiosa experiencia, consejos y sabiduría durante mi formación. Agradezco el apoyo por el proyecto DGAPA PAPIIT IN114924. Al Instituto de Ciencias de la Atmósfera y Cambio Climático. Por prestarme la infraestructura y recursos para realizar este proyecto.

Quiero expresar mi profundo agradecimiento a mis amigos, quienes siempre me han impulsado a seguir mis sueños y metas. Han confiado en que puedo lograrlo y me han alentado en todo momento. Sus palabras de apoyo, admiración y reconocimiento a mi trabajo han sido un impulso fundamental para seguir adelante y progresar.

Resumen

En esta tesis se desarrolla una maqueta de pruebas con máquinas virtuales para implementar tokens como método de autenticación, un medio de autorización de usuarios en un entorno Grid. Esta implementación surge debido a la necesidad de sustituir los certificados x509 por el uso de tokens en la versión 8.9.2 del software HTCondor [1]. Este trabajo mostrará al lector un panorama general de la importancia del cómputo Grid en América, así como algunos conceptos importantes para entender la Grid.

Una maqueta de pruebas representa una pequeña fracción del gran trabajo que se está desarrollando en nuestra Universidad para obtener una infraestructura Grid.

La Grid UNAM es una iniciativa que responde al Programa 3.2 de Investigación e Innovación del Plan de Desarrollo Institucional 2019-2023, específicamente al proyecto 7, denominado “Actualizar y consolidar la infraestructura de cómputo orientada a la investigación y capacitar al personal académico de todos los campos en materia de acceso y operación de los recursos existentes”. [2]
En este proyecto participan: La dirección general de cómputo y tecnologías de información y comunicación, Instituto de Ciencias Nucleares, Instituto de Astronomía, Instituto de Ciencias de la Atmósfera y Cambio Climático.

Contenido:

Capítulo 1: Se define el entorno grid, en redes académicas y se mencionan algunos ejemplos de grids.

Capítulo 2: Se explica y justifica el motivo de cambio de tecnología x.509 a tokens y se describen los objetivos del proyecto.

Capítulo 3: Se definen los conceptos que son imprescindibles para entender el contexto detrás de este trabajo.

Capítulo 4: Se explican los pasos para el desarrollo de la maqueta.

Capítulo 5: Se exponen las pruebas de autorización y envío de trabajos.

Capítulo 6: Se comentan las conclusiones.

Capítulo 7: Se mencionan posibles desarrollos a futuro.

Anexos: Se agrega información complementaria utilizada en la realización del proyecto.

Glosario:

TÉRMINOS	Descripción
DHTC	Distributed High Throughput Computing. Computación Distribuida de Alto Rendimiento
GSI	GRID SECURITY INFRASTRUCTURE. Una especificación para comunicación segura en Grids usando encriptación asimétrica.
IDP	Provider ID. Proveedor de identidad. Almacena y gestiona las identidades digitales de los usuarios.
JWT	Json web token Es un estándar abierto (RFC 7519) para transferir de forma segura información entre partes.
OAuth	Open Authorization. Es un protocolo de autorización que permite a una aplicación obtener acceso a recursos protegidos en nombre de un usuario sin compartir sus credenciales de autenticación.
SciAuth	Es un marco de autorización y autenticación específico para aplicaciones y servicios científicos que utilizan infraestructuras de computación distribuida.
PKI	Public Key Infrastructure. Un grupo de componentes y servicios informáticos que permiten administrar, gestionar, controlar la tarea de generar, brindar, revocar y validar toda clase de certificados digitales.
CA	Certification Authority. Entidad de confianza que emite y gestiona certificados digitales.
RFC	Request for Comments. Publicaciones de los principales organismos técnicos encargados de los estandares y buenas prácticas de internet.
GRID	GRID Es una infraestructura de cómputo distribuido que permite compartir recursos de cómputo
HTC	High Throughput Computing. Es el Cómputo de alto rendimiento, permite un gran número de cálculos seriales o de memoria compartida. Es decir cálculos independientes o poco acoplados.
HPC	High Performance Computing. Es el Cómputo de alto rendimiento utilizado para ejecutar trabajos paralelos muy acoplados que requieren un gran poder de cálculo intensivo.
SLURM	Es un gestor de tareas comúnmente usado en HPC.
HTCondor	Es un sistema de software que crea un entorno HTC. Utiliza de forma eficaz la potencia informática de las máquinas conectadas en una red.
IETF	Internet Engineering Task Force Organización encargada de establecer estándares abiertos para internet.
LIGO	Laser Interferometer Gravitational-Wave Observatory en EU. Instalación en Estados Unidos que utiliza interferómetros láser para detectar y estudiar ondas gravitacionales
OSG	Open Science Grid. Es un consorcio dedicado al avance de la ciencia Distributed High Throughput Computing (DHTC).
WLCG	Worldwide LHC Computing GRID. Infraestructura de computación global que respalda las necesidades de procesamiento y análisis de datos de las pruebas en el Gran Colisionador de Hadrones “LHC” alrededor del mundo con cerca de 170 centros en más de 40 países.

NREN	"National research and education network" Se refiere a una red nacional destinada a la investigación y educación, que se establece en un país específico para proporcionar conectividad y servicios de red avanzados.

Redes y Proyectos Grid

TÉRMINO	*Descripción*
ALICE	"América Latina Interconectada Con Europa" Infraestructura de conectividad para la investigación en América Latina, interconectada con Europa.
BELLA	"Building the Europe Links with Latin America" Iniciativa que tiene como objetivo crear una infraestructura de conectividad entre Europa y América latina.
CUDI	Corporación Universitaria para el Desarrollo de Internet Es una asociación civil sin fines de lucro que gestiona la Red Nacional de Investigación y Educación (RNIE) en México.
EELA	"E-infrastructure Shared between Europe and Latin America" Infraestructura de red común entre Europa y América Latina.
ESnet	Energy Sciences Network Red de infraestructura de alto rendimiento que beneficia a la comunidad de investigación científica dentro del departamento de Energía de los Estados Unidos.
GISELA	"Grid Initiatives for e-Science virtual Communities en Europa y América Latina" Iniciativa Grid Latinoamericana con asociación con RedClara, con la finalidad de proporcionar infraestructura, servicios.
Internet2	Internet2, también conocido como UCAID (University Corporation for Advanced Internet Development) Se trata de una comunidad que ofrece una infraestructura de red segura, de gran velocidad, soluciones en la nube, asistencia a la investigación y servicios a medida para investigación, educación
Pragma	"Pacific Rim Application and Grid Middleware Assembly" Es una comunidad colaborativa del Pacífico RIM que ayuda a grupos pequeños y medianos a resolver problemas utilizando tecnología de la información.
REDCLARA	Cooperación Latino Americana de Redes Avanzadas. Organismo internacional sin fines de lucro que brinda Infraestructura de Red**,** interconecta Redes Nacionales de Investigación y Educación (RNIE) de países latinoamericanos.
Arpanet	"Advanced Research Projects Agency Network" Fue una red académica en Estados Unidos, 1969.

Índice General

1. Introducción: Antecedentes del entorno Grid y las redes académicas

En este capítulo se definirán y explicarán conceptos como Grid, Redes académicas y la relación.

1.1. Tecnología Grid

El término Grid fue utilizado a mediados de los años 90´s. Es una infraestructura de cómputo distribuido, la cual permite a distintas instituciones y organizaciones compartir recursos de cómputo como: almacenamiento, procesamiento y aplicaciones específicas. Este tipo de infraestructura descentralizada, conecta recursos heterogéneos (arquitectura, cluster y supercomputadoras) mediante redes de área extensa como internet.

Posteriormente al desarrollo de las primeras computadoras con capacidad de conexión a internet. Surgió la idea de aprovechar la potencia a menudo inutilizada de estos equipos, con la finalidad de abordar problemas que exigen mayor capacidad de procesamiento y almacenamiento. Estos problemas, que solían ser exclusivamente tratados por supercomputadoras de uso académico, gubernamental o privado. A partir de esta necesidad se da origen a la llamada Computación GRID (acrónimo de Global Research infrastructure for Data), se refiere a una infraestructura global de investigación para el manejo y el intercambio de datos científicos a gran escala.

El GRID es un sistema que facilita la colaboración, el acceso a recursos y servicios de datos distribuidos en diferentes instituciones y ubicaciones geográficas. Cuyo objetivo es el aprovechamiento de equipos subutilizados que están conectados a una red.

Una definición práctica la encontramos en la tesis llamada “Computación Grid” en la página 21.
“Se considera entonces, que la Computación Grid es una infraestructura que involucra hardware y software capaz de brindar al usuario un acceso seguro, consistente, penetrante, a bajo costo, y con grandes capacidades computacionales”[3]. GRID es una herramienta valiosa en el ámbito científico y de Investigación, ya que permite aprovechar de manera eficiente los recursos y promover la colaboración y el avance científico a nivel global.

Entre los beneficios de este tipo de infraestructura se encuentra:

- Utiliza y optimiza el uso de los recursos existentes.
- Brinda mayor poder de cómputo.
- Bajo costo.
- Incrementa la productividad.
- Facilita el retorno de inversión.
- Abre la posibilidad de compartición y participación en nuevos proyectos.
- Evita cuellos de botella, mejora los tiempos de respuesta y garantiza mayor disponibilidad de recursos
- Flexibilidad.
- Utilización de <software libre y de código abierto>.
- En el ámbito académico, constituye una gran herramienta para apoyar a los proyectos de investigación.
- Da capacidad de cómputo a quienes no lo tienen a un costo institucional reducido.
- Promueve la colaboración y el intercambio de conocimientos entre investigadores y científicos de diversas instituciones y países.
- Proporciona mayor disponibilidad y fiabilidad de los recursos, ya que la infraestructura está diseñada para tolerar fallos y garantizar la continuidad del servicio.
- Promueve el uso de estándares abiertos y protocolos de comunicación, lo que facilita la interoperabilidad entre diferentes sistemas y plataformas.

1.2. Redes Académicas

El concepto de redes académicas surge a partir de ARPANET Advanced Research Projects Agency Network, con el respaldo del Departamento de Defensa de los Estados Unidos y la NSF en 1969. [4] ARPANET Fue creada como un medio de comunicación entre instituciones académicas y gubernamentales en Estados Unidos. Debieron pasar muchos años para que surgiera INTERNET 2 la cual es una red académica avanzada separada desde sus inicios de la internet comercial. INTERNET 2 nace en 1996, con el propósito de proporcionar un entorno de red con velocidades mucho mayores y capacidades más avanzadas que las redes comerciales en ese momento, ofreciendo una amplia gama de recursos y herramientas para la investigación y la educación en los Estados Unidos. Llegó a México en el año 2002.

Las redes académicas avanzadas son redes de computadoras diseñadas específicamente para ofrecer servicios en beneficio de la investigación, desarrollo y comunicación de Universidades, centros e institutos. Consisten en un conjunto de redes diseñadas para proporcionar un alto rendimiento con anchos de banda que van desde los 2 Mbps hasta los 10 Gbps, aunque actualmente existen redes como ESnet “Energy Sciences Network” que es la red de ciencias energéticas del departamento de Energía de Estados Unidos. Esta red cuenta con velocidades promedio de cientos de Gbps [5]. Es considerada la red académica más veloz hasta el momento. En la imagen 1 podemos darnos una idea de la carga de trabajo en ESnet.

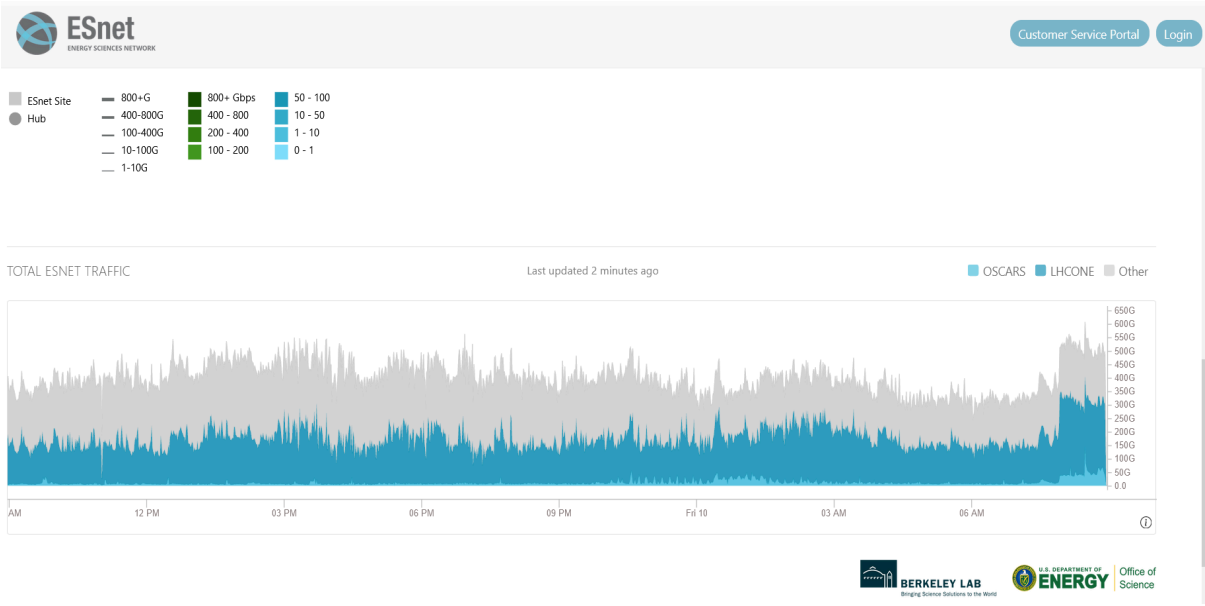

Imagen 1: Tráfico total en la red ESnet.

Muchos países tienen sus Redes Nacionales de Educación e Investigación (RNEI), conocidas por sus siglas en inglés como NREN “National Research and Education Network”. Cuando algunas de estas redes NREN se conectan, forman redes continentales. Es aquí donde el concepto de redes académicas es potenciado e inician las colaboraciones Internacionales y surgen proyectos como ALICE, REDCLARA, EELA entre otros.

1.3. Proyectos Grid y Redes académicas en América

Proyectos como Internet2, ALICE, EELA, RedClara, Cudi, PRAGMA entre otros son el punto de unión entre una NREN y el poder de cómputo grid para apoyar el desarrollo científico en el mundo.

1.3.1. ALICE, ALICE-2

Entre los años 2003 y 2008, se desarrolló el proyecto América Latina Interconectada con Europa (ALICE), como se indica en la ver tabla 1, con el objetivo principal de establecer una infraestructura de redes de investigación en América Latina y conectarla con su contraparte europea. [6]
Dentro de los países participantes estaban con sus respectivas redes académicas:

País	Red Académica	Nombre completo
España	RedIRIS	Red para la Interconexión de los Recursos InformáticoS
Francia	RENATER	Réseau national de télécommunications pour la technologie
Italia	GARR	Gruppo per l'Armonizzazione delle Reti della Ricerca
Portugal	FCCN	Fundação para a Computação Científica Nacional
Argentina	RETINA	Red Teleinformática Académica
Bolivia	ADSIB	Agencia para el desarrollo de la Sociedad de la Informática en Bolivia.
Brasil	RNP	Rede Brasileira para educação e pesquisa
Chile	REUNA	Red Universitaria Nacional
Colombia	Universidad del Cauca	Universidad del Cauca
Costa Rica	CRnet	Red Nacional de Investigación de Costa Rica
Cuba	RedUniv	Red universitaria de Cuba
Ecuador	CEDIA	Corporación Ecuatoriana para el Desarrollo de la Investigación y la Academia.
El Salvador	RAICES	Red Nacional de Investigación y Educación de El Salvador
Guatemala	RAGIE	Red Avanzada Guatemalteca para la Investigación y Educación
Honduras	UNITEC	Universidad Tecnológica Centroamericana

México	**CUDI**	**Corporación Universitaria para el Desarrollo de Internet**
Nicaragua	**CNU**	**Consejo Nacional de Universidades**
Panamá	**RedCyT**	**Red Científica y Tecnológica**
Paraguay	**ARANDU**	**Arandu palabra guaraní significa: conocimiento, sabiduría.**
Perú	**RAP**	**Red académica Peruana**
Uruguay	**RAU**	**Red Académica Uruguaya**
Venezuela	**REACCIUN**	**Red Académica de Centros de Investigación y Universidades Nacionales**

Tabla 1: Países y Redes académicas Participantes en el proyecto Alice.

El proyecto ALICE-2 se llevó a cabo desde 2008 hasta 2013 con el fin de ampliar y reforzar la infraestructura de la RedCLARA para la investigación. Para lograr este objetivo, se lanzó el proyecto ALICE2, que comenzó a funcionar en el primer trimestre de 2009.

Su objetivo fue: "Fomentar y apoyar la investigación colaborativa dentro de América Latina y entre la región y Europa, a través del fortalecimiento de CLARA (Cooperación Latinoamericana de Redes Avanzadas) y su infraestructura de red (RedCLARA), la promoción de la creación y mantenimiento de comunidades de investigación que trabajen en temas relacionados con el desarrollo" [7].

1.3.2. Bella: Building the Europe Links with Latin America

Bella fue un proyecto de red que tuvo lugar entre los años 2016-2021. [8] Su objetivo era proporcionar apoyo para la interconexión a largo plazo de las comunidades de investigación, educación europeas y latinoamericanas.
Bella está compuesto de 11 redes de investigación, educación europeas y latinoamericanas, entre ellas: Brasil, Chile, Ecuador, Alemania, España, Francia, Italia y Portugal.

1.3.3. CUDI: Corporación Universitaria para el Desarrollo de Internet

Fundada en 1998 como una Asociación CIVIL sin fines de lucro dedicada a instituciones de educación superior e investigación. CUDI es una entidad sin ánimo de lucro que administra la **Red Nacional de Educación e Investigación** (RNEI). Actualmente más de 130 países cuentan con una RNIE, a nivel mundial este tipo de redes se interconectan para formar una red académica avanzada como se muestra en la imagen 2 . CUDI es una RNIE, en su red troncal nacional de 10 Gbps se encuentran más de 505 instituciones académicas conectadas. [9]

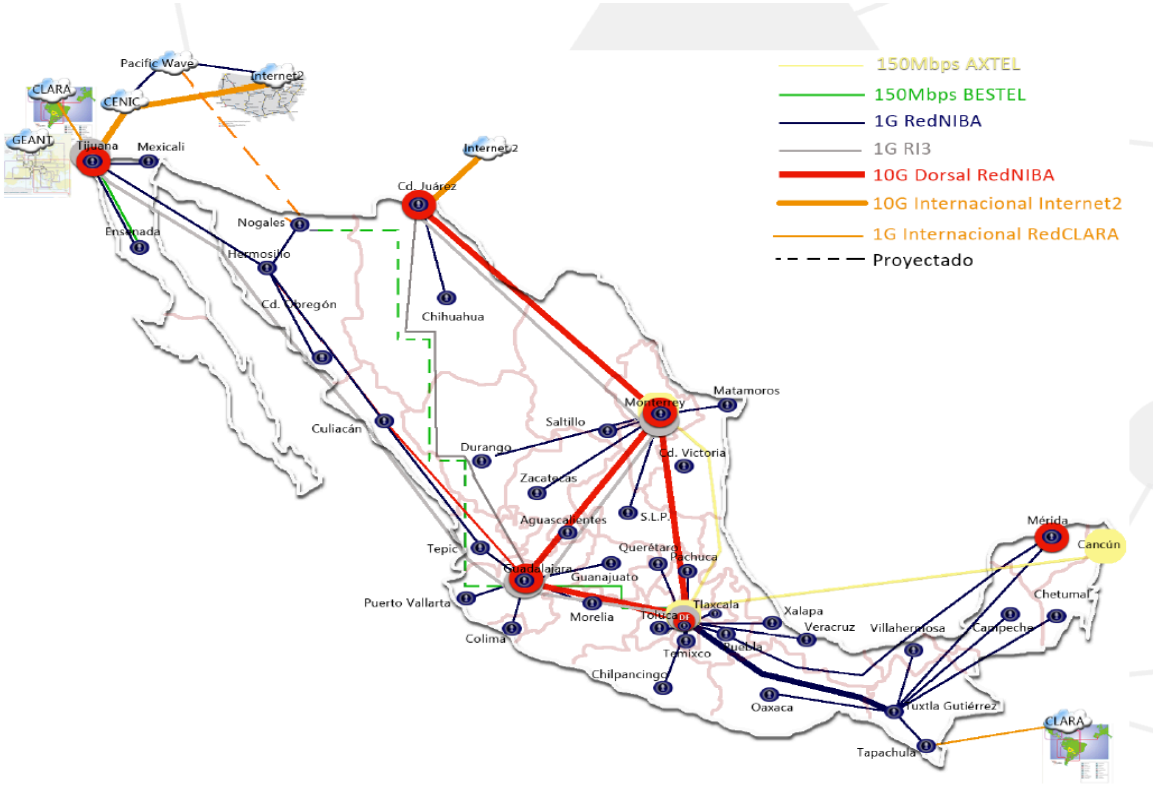

Imagen 2: Infraestructura CUDI.

1.3.4. EELA, EELA-2

El proyecto EELA, también conocido como “E-Infrastructure shared between Europe and Latin America”, se comenzó el 1 de enero de 2006 y tuvo una duración de 24 meses. Fue coordinado por CIEMAT en España y ejecutado por 21 instituciones de Europa y América Latina durante este periodo. El propósito de EELA consiste en elevar el nivel de las e-Infraestructuras de los países de América latina al nivel de los de Europa. “Su enfoque de EELA es crear una red de colaboración que se encargará de la organización de la formación de tecnologías Grid y del despliegue de una infraestructura Grid piloto para aplicaciones de E-ciencia” [10]

El proyecto EELA-2, también conocido como "E-Infrastructure shared between Europe and Latin America 2", tuvo lugar entre el 1 de enero de 2008 y 2010.

Su objetivo principal era crear una red computacional de alta capacidad, calidad y escalabilidad, que ofreciera acceso global a recursos de cómputo distribuido, almacenamiento y red. Este sistema estaba destinado a soportar una amplia gama de aplicaciones utilizadas en colaboraciones científicas entre Europa y América Latina. Se ponía especial énfasis en proporcionar una variedad completa de servicios versátiles para satisfacer las necesidades de estas aplicaciones y asegurar la sostenibilidad a largo plazo de la infraestructura electrónica más allá de la finalización del proyecto.

Los países que participaron en este proyecto fueron: España, Cuba, Latino America (RedCLARA), Irlanda, Francia, Italia, Argentina, Chile, Venezuela, Colombia, México, Portugal, Brasil, Ecuador, Brasil.
Se construyó sobre las redes de investigación GÉANT y RedCLARA. Ambas son NRENs de Europa y latino America respectivamente, para convertirse en un recurso e-Infraestructura que ofrezca un conjunto de servicios para diversas aplicaciones en múltiples áreas científicas, beneficiando a las comunidades científicas de Europa y Latino América". [11]

1.3.5. ESnet: Energy Sciences Network

Se formó oficialmente en 1986, aunque sus raíces se remontan a mediados de la década de 1970. Actualmente es considerada una de las redes académicas más veloces del mundo. Esnet proporciona conexiones confiables con gran ancho de banda que vinculan a los científicos de los laboratorios Nacionales, universidades y otras instituciones de investigación ver imagen 3.
Algunos temas que se abordan gracias a esta red son: Energía, ciencia del clima y orígenes del universo.
La misión de Esnet es: proporcionar el tipo especializado de servicios de red requeridos por equipos multidisciplinarios de científicos mientras colaboran en experimentos distribuidos utilizando equipos científicos, supercomputadoras e instalaciones de clase mundial. [12]

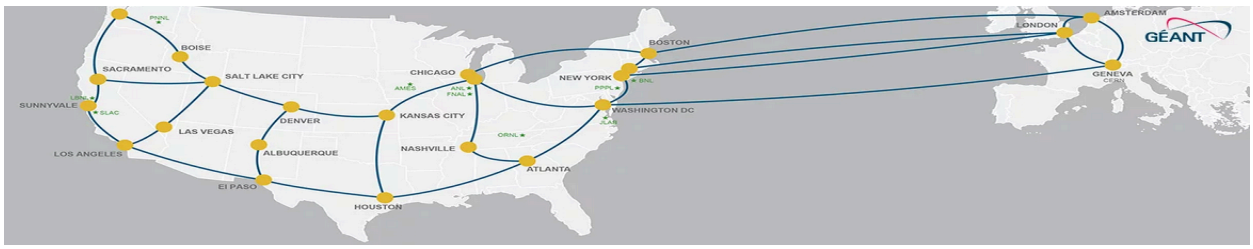

Imagen 3: Infraestructura ESnet.

1.3.6. GISELA: Grid Initiatives for e-Science virtual Communities en Europa y América Latina.

GISELA, cuyo nombre completo en inglés es "Grid Initiatives for e-Science virtual Communities in Europe and Latin America", fue un proyecto llevado a cabo desde el 21 de septiembre del 2010 hasta el año 2012. Su objetivo principal fue desarrollar un modelo de sostenibilidad para la iniciativa Grid Latinoamericana, apoyándose en proyectos Grid Nacionales, en colaboración con REDCLARA, las Redes

Nacionales de Investigación y Educación de América Latina, junto con la Iniciativa Grid Europea. Todo ello con el propósito de proporcionar a las comunidades virtuales de investigación una infraestructura, herramientas y servicios que mejoraran la eficacia de sus investigaciones.

Dentro de los países participantes encontramos: España, Ecuador, Panamá, Francia, Cuba, México, Italia, Argentina, Chile, Uruguay, Brasil, Venezuela, Colombia, Portugal y Polonia. [13]

1.3.7. Internet 2

Internet2, también conocida como UCAID "*University Corporation for Advanced Internet Development*", es un consorcio sin ánimo de lucro que opera la red internet2 y fue establecido en 1996. Surgió cuando aproximadamente 34 Universidades de los Estados Unidos se unieron con el objetivo de crear un proyecto capaz de impulsar la creación de una infraestructura de investigación y educación.
Internet2 es una red informática avanzada separada del internet comercial. Ofrece una conexión segura y de alta velocidad, soluciones basadas en la nube, apoyo a la investigación y servicios personalizados para la investigación y la educación. [14]
Enlaza universidades en todo el mundo para el desarrollo de sofisticadas aplicaciones. "Algunas de las aplicaciones en desarrollo dentro del proyecto de Internet 2 a nivel internacional son: telemedicina, bibliotecas digitales, laboratorios virtuales, manipulación a distancia y visualización de modelos 3D; aplicaciones todas ellas que no serían posibles de desarrollar con la tecnología del Internet de hoy." [15]

1.3.8. PRAGMA

PRAGMA "Pacific Rim Application and Grid Middleware Assembly" es una comunidad de práctica internacional y distribuida, compuesta por personas e instituciones de todo el Pacific Rim que colaboran activamente con grupos pequeños y medianos lo cual les permite resolver sus problemas con la tecnología de información. [16]

1.3.9. RedClara

RedClara [17] es una red académica que conecta las Redes Nacionales de Investigación y Educación (RNIE) de los países de América Latina. Después de la formación de ALICE, 13 países latinoamericanos se unieron para crear la Cooperación Latinoamericana de Redes Avanzadas (CLARA), y desde entonces cada país miembro ha ido estableciendo su propia Red Académica Nacional, Actualmente, 12 de estas redes están conectadas a la redCLARA, y otras en proceso de desarrollo.

Desde el año 2004, RedCLARA ha proporcionado conexiones regionales y globales a través de sus enlaces con GÉANT (la red avanzada paneuropea) e Internet2. A través de estas conexiones, RedCLARA se vincula con las redes avanzadas en África (como UbuntuNet Alliance, WACREN, ASREN), Asia (incluyendo APAN, TEIN, CAREN) y Oceanía (AARNET), entre otras. (ver imagen 4).

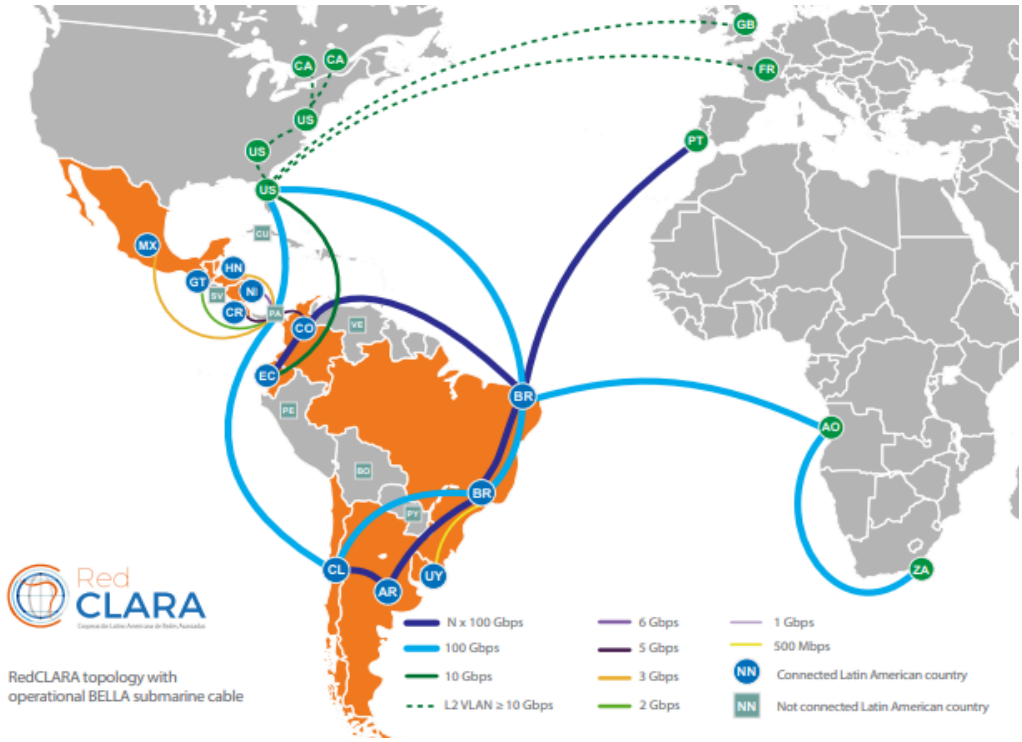

Imagen 4: Topología de la red clara.

1.3.10. RINGRID: Remote Instrumentation on Next Generation Grids

El proyecto RINGrid, cuyo traducción de su nombre completo es "Instrumentación Remota en Mallas Computacionales de Próxima Generación", comenzó el 1 de octubre del 2006 y concluyó en 2008. El objetivo principal de este proyecto fue validar y establecer estándares para la utilización de instrumentación remota en redes de mallas computacionales (Grids).
Los Países participantes fueron: Polonia, Austria, Grecia, Bulgaria, México, Italia>, AmericaLatina, Chile, Brasil, Reino Unido y Rumania. [18]

Con el fin de tener una visión general de los proyectos abordados en el capítulo 1, se puede hacer referencia a la Tabla 2 y a la Imagen 5. La Tabla 2 presenta una clasificación de los proyectos Grid y Redes Académicas, mientras que la Imagen 5 muestra una línea de tiempo de los mismos proyectos.

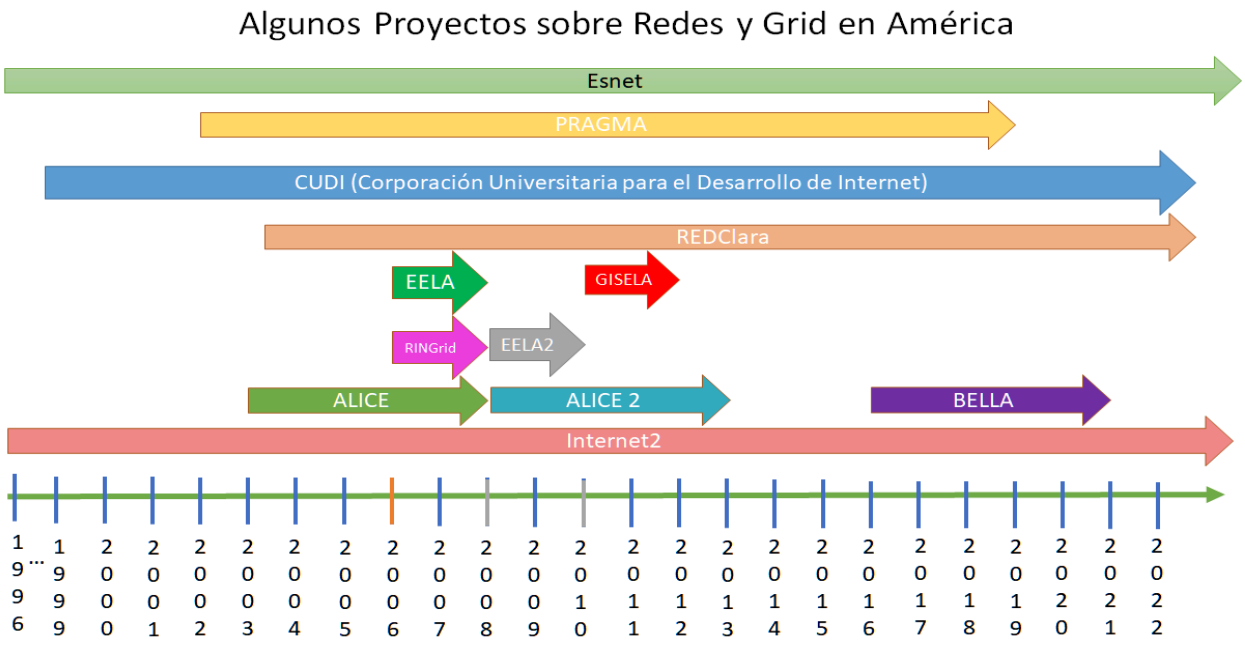

Imagen 5: Línea de tiempo de los proyectos Grid y Redes académicas del capítulo 1.

TÉRMINO	Descripción	Tipo Enfoque
ALICE	América Latina Interconectada Con Europa	RED
BELLA	Building the Europe Links with Latin America	RED
CUDI	Corporación Universitaria para el Desarrollo de Internet	**RED**
EELA	E-infrastructure Shared between Europe and Latin America	**RED**
ESnet	Red de ciencias energéticas del departamento de Energía de Estados Unidos.	RED
GISELA	Grid Initiatives for e-Science virtual Communities en Europa y América Latina.	GRID
PRAGMA	Pacific Rim Application and Grid Middleware Assembly	GRID
REDCLARA	La red avanzada de América latina.	RED
RINGrid	Remote Instrumentation on Next Generation Grids	GRID
UCAID	University Corporation for Advanced Internet Development también conocido como Internet2	RED

Tabla 2: Clasificación de Proyectos Grid y Redes Académicas.

Ahora que tenemos más en claro el contexto e importancia de estos dos conceptos: Redes académicas y cómputo Grid.

2. Descripción del problema

En el presente capítulo se presentan los motivos tecnológicos e históricos que impulsaron el desarrollo de este trabajo, así como su importancia y justificación.

2.1. Grid UNAM

Históricamente, diversas entidades de la UNAM han realizado grandes esfuerzos para disponer de infraestructura de cómputo de alto rendimiento, beneficiando el desarrollo de proyectos científicos. Sin embargo en muchas ocasiones los recursos computacionales adquiridos por dichas entidades no resultan ser suficientes incluso para el desarrollo de sus propios proyectos. Por otra parte, aún existen otras entidades académicas que no poseen infraestructura y poderío de cómputo para el desarrollo de sus investigaciones y trabajos científicos. Esta es la principal razón por la cual surge el proyecto “Grid UNAM”.

Grid Unam tiene por objetivo apoyar proyectos de investigación que necesitan una infraestructura de cómputo de alto rendimiento con mayores capacidades.

Actualmente la UNAM cuenta con una variedad de centros de cómputo de alto rendimiento (HPC, por sus siglas en inglés) y centros de almacenamiento de Datos distribuidos en varios campos y ubicaciones dentro de la misma Universidad. Cada uno cuenta con diferentes capacidades, tanto en volumen de procesamiento, almacenamiento, número de usuarios y personal especializado.
Este hecho hace ocasiona que:

- Centros de cómputo se encuentran sobresaturados sin posibilidad de ofrecer más servicios, lo cual provoca tener usuarios en espera, mientras otros centros de cómputo tienen disponibilidad de ejecutarlos de manera temporal.
- Existen centros de cómputo que solo registran saturación en ciertos momentos, siendo en otros momentos subutilizados.
- Existen aplicaciones que requieren de capacidades que no se encuentran en los HPC de sus entidades, mientras que otras entidades podrían procesar dichas tareas.

El proyecto Grid Unam tiene como objetivo compartir de manera colaborativa los recursos de procesamiento, cómputo y almacenamiento, aprovechando la infraestructura existente. Actualmente se encuentra en etapas de desarrollo, implementación, pruebas y producción.

La maqueta que se presenta en esta tesis refleja solamente una parte pequeña del proyecto. Este trabajo se enfoca en la autorización de usuarios a recursos de la Grid mediante el uso de Tokens y envíos de trabajos htcondor-ce.

Inicialmente, con el software HTCondor se utilizaban certificados para autorizar a los usuarios en el uso de recursos. Sin embargo, la versión 8.9.2 del software HTCondor marcó un cambio importante en el proceso de autenticación y autorización de usuarios dentro del entorno de Grid. En lugar de utilizar certificados X.509, se implementaron Json Web Tokens. Este cambio trajo consigo términos como Tokens, Json Web Tokens, OpenID, Oauth entre otros. Dichos términos cobran relevancia para entender la implementación y uso de este tipo de autenticación, autorización por medio de tokens.

Esta migración inició en 2018 cuando el proyecto "Globus project" terminó su soporte para "Globus Toolkit" y su GRID SECURITY INFRASTRUCTURE (GSI) basada en X.509.
En ese mismo año la NSF (National Science Foundation) financia el proyecto "SciTokens", el cual demostró ser viable para hacer esta migración de certificados GSI a JSON Web Tokens (JWT) un estandar del IETF (ver imagen 6) [19]. Esta migración ha facilitado un replanteamiento de la autenticación y autorización entre los proveedores de ciber infraestructuras, trayendo consigo ventajas en la seguridad como implementar un sistema de mínimos privilegios que se basa en la capacidad. [19]

Para 2019 el Worldwide LHC Computing GRID (WLCG) adoptó el modelo SciTokens. En 2020, 11 proyectos Open Science Grid utilizaron JWTs con SciTokens para más de 115,000 transferencias de datos científicos.

OSG planea retirar GSI en 2022 y WLCG planea comenzar a **eliminar los certificados de usuario X.509** de su infraestructura en 2023. *Está es una de las razones más importantes para adoptar e implementar de forma inmediata esta tecnología.* Integrar un nuevo mecanismo de seguridad en una ciberinfraestructura de producción lleva años.

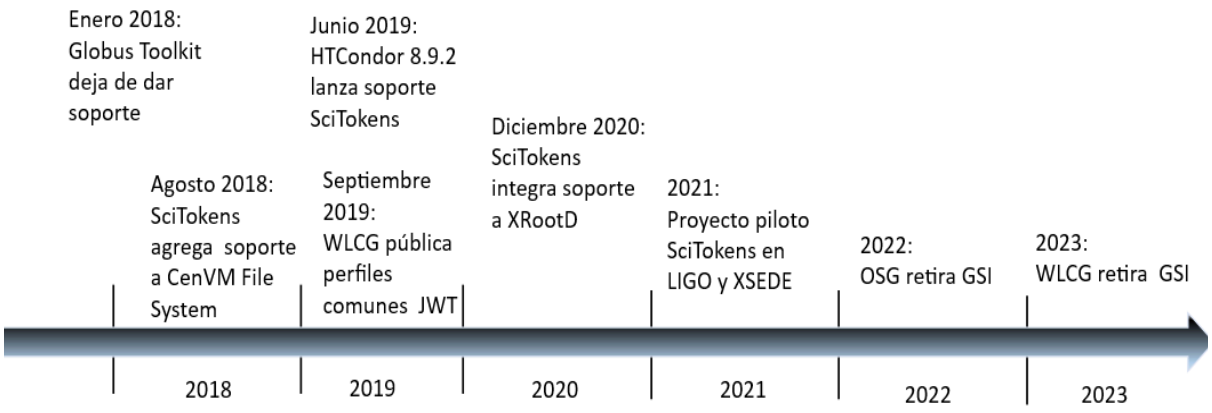

Imagen 6: línea de tiempo de la adopción de Scitokens.

2.2. Justificación

La implementación de tokens como método de autenticación en el proyecto corresponde al hecho de retirar los certificados de usuario x.509 en el entorno grid. Por otra parte es una elección estratégica respaldada por varias razones fundamentales:

1. Mayor seguridad: Los tokens pueden ser encriptados[1] y firmados digitalmente[2],lo que garantiza la confidencialidad e integridad de la información transmitida. Además, los tokens son específicos del usuario y suelen tener una fecha de vencimiento, lo que reduce el riesgo de uso indebido.

2. Mayor flexibilidad: Los tokens son portables y pueden ser utilizados en diferentes contextos y aplicaciones, lo que brinda una mayor flexibilidad en la autenticación de usuarios en diferentes sistemas o servicios. Esto permite una integración más fácil y rápida en entornos heterogéneos y simplifica la gestión de identidad en sistemas distribuidos o en la nube.

3. Mejora de la experiencia del usuario: La autenticación basada en tokens puede ofrecer una mejor experiencia para los usuarios finales. Los tokens son menos intrusivos en términos de solicitar información confidencial, contraseñas, en comparación con otros métodos de autenticación. Esto puede resultar en un proceso de inicio de sesión más sencillo y conveniente para los usuarios, lo que a su vez puede mejorar la adopción y aceptación del sistema.

4. Integración con estándares modernos: Los tokens son ampliamente utilizados en la actualidad como parte de estándares modernos de autenticación y autorización, como OAuth, OpenID Connect, y JWT (Json Web Tokens). Por otra parte, muchos de los lenguajes más utilizados brindan librerías para su implementación. Utilizar tokens en el proyecto nos permite seguir las buenas prácticas así como los estándares de la industria, lo que puede facilitar la interoperabilidad con otros sistemas y servicios.

[1] La encriptación se refiere al proceso de transformar la información del token en un formato ilegible para cualquiera que no tenga la clave de descifrado correspondiente.

[2] Una firma digital es un proceso criptográfico que garantiza la autenticidad e integridad de un token. Una firma digital es un código que garantiza la autenticidad de un token.

2.3. Objetivos

2.3.1. Objetivo Principal:

Autorización de usuarios por medio de tokens en un entorno Grid. Para lo cual se desarrollará una maqueta con la capacidad de utilizar tokens, enviar trabajos a un cluster y administrar usuarios. Esta maqueta cuenta con tres nodos de procesamiento, un nodo maestro, un nodo submit y un nodo iam. Los sistemas operativos usados son Centos 7 y Debian 11. Algunos de los softwares a utilizar son: HTCondor-CE, Slurm, Indigo IAM, OpenID Connect, Munge.

2.3.2. Objetivos Secundarios:

1) Investigar, examinar los siguientes conceptos: Redes Académicas, Tecnología Grid, HTC, HPC, JWT, X509, OAuth, INDIGO, OIDC.

2) Establecer una infraestructura virtual de un Clúster: Nodo maestro, Nodos de procesamiento.

3) Instalar y configurar la siguiente lista de software en dicha infraestructura:
    A. Configuración de la red en la infraestructura.
    B. Instalar y configurar Munge.
    C. Instalar, configurar y hacer pruebas Slurm.
    D. Compartir cuentas de usuarios en la configuración del cluster.
    E. Compartir algunas carpetas entre los nodos.
    F. Instalar y configurar HTCondor-CE.
    G. Instalar y configurar Indigo.
    H. Instalar, configurar y hacer pruebas OIDC.
    I. Obtener, utilizar Tokens como método de autorización.
    J. Mandar trabajos de prueba a la grid usando tokens.

3. Marco teórico

En este capítulo se presentan los conceptos teóricos fundamentales necesarios para comprender y desarrollar el proyecto de Autorización de usuarios mediante tokens en un entorno Grid. Se proporciona una introducción detallada y explicación de temas como el estándar X.509, JSON Web Token, HPC, HTC, Htcondor, Slurm, SSO, OAuth, OpenID e INDIGO. El orden en que se presentan los temas brinda una mejor comprensión de la relación existente entre los temas a medida de que se desarrollan.

3.1. Estándar x509

X.509 es una recomendación fundamental que establece un marco para la emisión y verificación de certificados de clave pública, los cuales son una herramienta crítica para garantizar la seguridad en entornos digitales. X.509 es ampliamente utilizado en la infraestructura de claves públicas [3] PKI.

X.509 no solo se relaciona a los certificados de llave pública (x.509), sino también define un marco para certificados de atributos, estos se utilizan para asociar información adicional con un certificado. La estructura definida por X.509 asegura que los certificados sean interoperables y confiables, lo que garantiza que los usuarios puedan verificar la identidad de los otros participantes en una comunicación segura en línea. A continuación definimos el concepto de norma, estándar y recomendación.

1. Una norma es una regulación o conjunto de reglas establecidas por una entidad gubernamental o una organización reconocida, con el propósito de estandarizar un proceso, producto o servicio en un determinado campo. En el ámbito de la informática, las normas son creadas por organismos reguladores y su cumplimiento puede ser obligatorio por ley.
2. Un estándar, por otro lado, es un conjunto de normas técnicas establecidas por una organización reconocida o consorcio de empresas, que describe las características, especificaciones y criterios de calidad que se deben cumplir para garantizar la interoperabilidad y compatibilidad de diferentes sistemas y tecnologías. A diferencia de las normas, el cumplimiento de los estándares no siempre es obligatorio, pero su adopción es ampliamente recomendada.
3. Una recomendación es un conjunto de directrices o mejores prácticas propuestas por una organización reconocida o grupo de expertos en un determinado campo, que no tienen carácter obligatorio, pero son altamente recomendadas para lograr un buen desempeño y calidad en el uso de tecnologías y sistemas informáticos.

“La recomendación UIT-T X.509 Se aprobó el 31 de marzo del año 2000 y como muchas recomendaciones ha sido elaborada para facilitar la interconexión de los sistemas de procesamiento de información con el fin de proporcionar servicios de directorio”. [20]

[3] PKI “Public Key Infrastructure”. Es un sistema que abarca hardware, software, políticas y procesos utilizados para gestionar certificados digitales (creación, distribución y revocación).

3.1.1. Certificados X.509

Los certificados X.509[4] son una aplicación del estándar x.509. Propuesto por la unión internacional de telecomunicaciones. [21] Esta clase de certificados x.509 representan una forma de certificados de clave o llave pública. Estos certificados cuentan con una firma digital e información como:

- Contiene información relevante a quien firmó el certificado
- Las clave pública.
- versión del certificado.
- Contiene información relevante a quien emite el certificado.
- También especifica el algoritmo de cifrado.
- Información relevante a la validez del certificado.

3.1.2. Versiones X.509

Existen 3 versiones: v1, v2 ,v3 de los certificados X.509 estas son incrementales ver imagen 7. la versión 1, la versión 2 y la versión 3.

La versión 1 es la más básica y solo contiene información mínima, como el nombre del titular del certificado y la clave pública correspondiente.

La versión 2 introduce extensiones para incluir información adicional, como políticas de uso de la clave y restricciones de uso.

La versión 3 es la más utilizada y ampliamente aceptada, ya que permite una mayor flexibilidad y complejidad en la inclusión de datos, como nombres alternativos del sujeto, políticas de certificación y más. En general en cada versión posterior se agregan nuevos campos ver tabla 3.

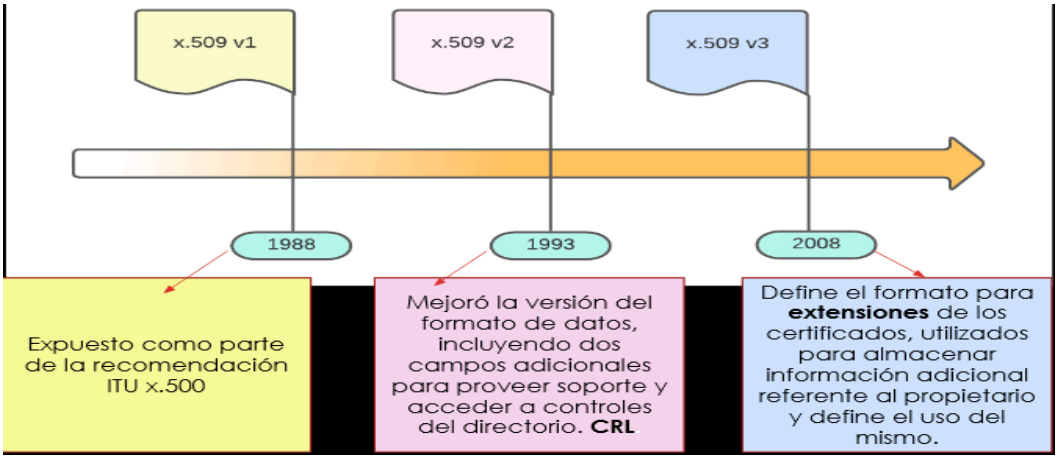

Imagen 7: Línea de tiempo de las diferentes versiones x.509.

[4] Certificado X.509: Es un documento digital que autentica la identidad (usuarios, dispositivos o servicios) mediante una llave pública, la cual fue emitida por una Autoridad Certificadora.

Los campos de las versiones son incrementales. Los campos de la versión 1 también se encuentran en la versión 2 más 2 campos adicionales como se muestra en la tabla 3. Por lo tanto la versión 3 contendrá los campos de las versiones anteriores.

Las tres versiones de los certificados X.509 son estándares clave en la PKI y han evolucionado para brindar mayor funcionalidad y seguridad en la emisión y gestión de certificados digitales.

X.509 Campos		
Versión 1	**Versión 2** **V2 = V1 + Campos**	**Versión 3** **V3=V2+Campos(Extensiones)**
Campo Versión: puede tener los valores: 1,2 o 3 según la versión del certificado.	**ID clave de la entidad emisora**: Es una clave única para la entidad emisora de certificados.	**Nombre alternativo de la autoridad**: Lista de nombres alternativos del firmante o autoridad emisora.
Número de serie: Identificador único que asigna la CA "Autoridad Certificadora"	**ID clave del sujeto**: Es una clave única opcional para identificar el sujeto del certificado.	**Identificador de clave de autoridad**: Clave pública correspondiente a la clave privada utilizada para firmar el certificado.
Algoritmo de firma CA: Algoritmo que usa la CA para firmar el certificado.		**Restricciones básicas**: Indica si el certificado es para un CA.
Nombre de emisor: Nombre distintivo "DN" de la CA que emite el certificado.		**Información de política del certificado**: Especifica las políticas bajo las cuales se emite el certificado.
Validez: El intervalo de tiempo durante que el certificado es válido.		**Nombre alternativo del sujeto**: Permite especificar nombre alternativos para el sujeto del certificado.
Nombre del firmante: Nombre distintivo "DN" del propietario del certificado.		**Restricciones de política**:Establece restricciones sobre el uso de políticas de certificado a seguir en la cadena de certificación.
Información de clave pública de quien firma: La clave pública del que firmó el certificado, localizada en el certificado.		**Otros:** ***Punto de distribución CRL** ***Directivas de certificado** ***Acceso a la información del firmante** ***etc…**

Tabla 3: Comparativa de los campos x.509 en sus diferentes versiones.

3.1.3. Seguridad X.509

X.509 es una recomendación, también conocida como ITU-T X.509, que define un estándar para el formato de certificados digitales. Este estándar ha sido modificado para su utilización en internet por la Infraestructura de clave pública X.509. Para comprender la infraestructura de llave pública (PKI) X.509 se necesitan comprender temas como algoritmos criptográficos, las claves[5] criptográficas, certificados y entidades CA. [22]

Entre las aplicaciones más comunes de los certificados X.509 se encuentran [23]:

- SSL[6]/TLS[7] y HTTPS utilizan los certificados para establecer conexiones seguras servidor-navegador.
- Firmado y cifrado de correos electrónicos mediante S/MIME.
- Firma de código para garantizar la autenticidad del software.
- Firma de documentos para validar la integridad y autenticidad de los archivos.
- Autenticación de clientes para acceso seguro a sistemas y servicios como el SAT en México.

El certificado es un documento digital que contiene la clave pública del dispositivo y se puede usar para verificar que el dispositivo es quien dice ser.

Actualmente, SSL ha sido ampliamente reemplazado por TLS, por lo que es probable que cualquier certificado utilizado por una página web tenga sea un certificado TLS. Un certificado TLS es un documento digital que verifica la identidad de un sitio web y permite una conexión cifrada.

Si un sitio web está protegido mediante un certificado TLS, en la URL aparecen las siglas HTTPS “el protocolo seguro de transferencia de hipertexto”. [24] Para la mejor comprensión de este tema se presenta el siguiente ejemplo donde se muestran algunos campos de un certificado TLS v1.3.

[5] Las **claves o llaves** son cadenas de datos aleatorios o pseudoaleatorios que se utilizan como entrada para un algoritmo.
[6] SSL “Secure Sockets Layer” fue desarrollado por Netscape en la década de 1990 y se convirtió en un estándar para proporcionar seguridad en las comunicaciones en línea. RFC 6101.
[7] TLS “Transport Layer Security” fue lanzado en 1999 como una actualización del protocolo SSL y su primera versión fue conocida como TLS 1.0. La versión actual de TLS es TLS 1.3, que fue lanzada en 2018. RFC 8446

Como ejemplo usaremos la página https://www.unam.mx. Una vez dentro de la página inspeccionamos el certificado (ver imagen 8).

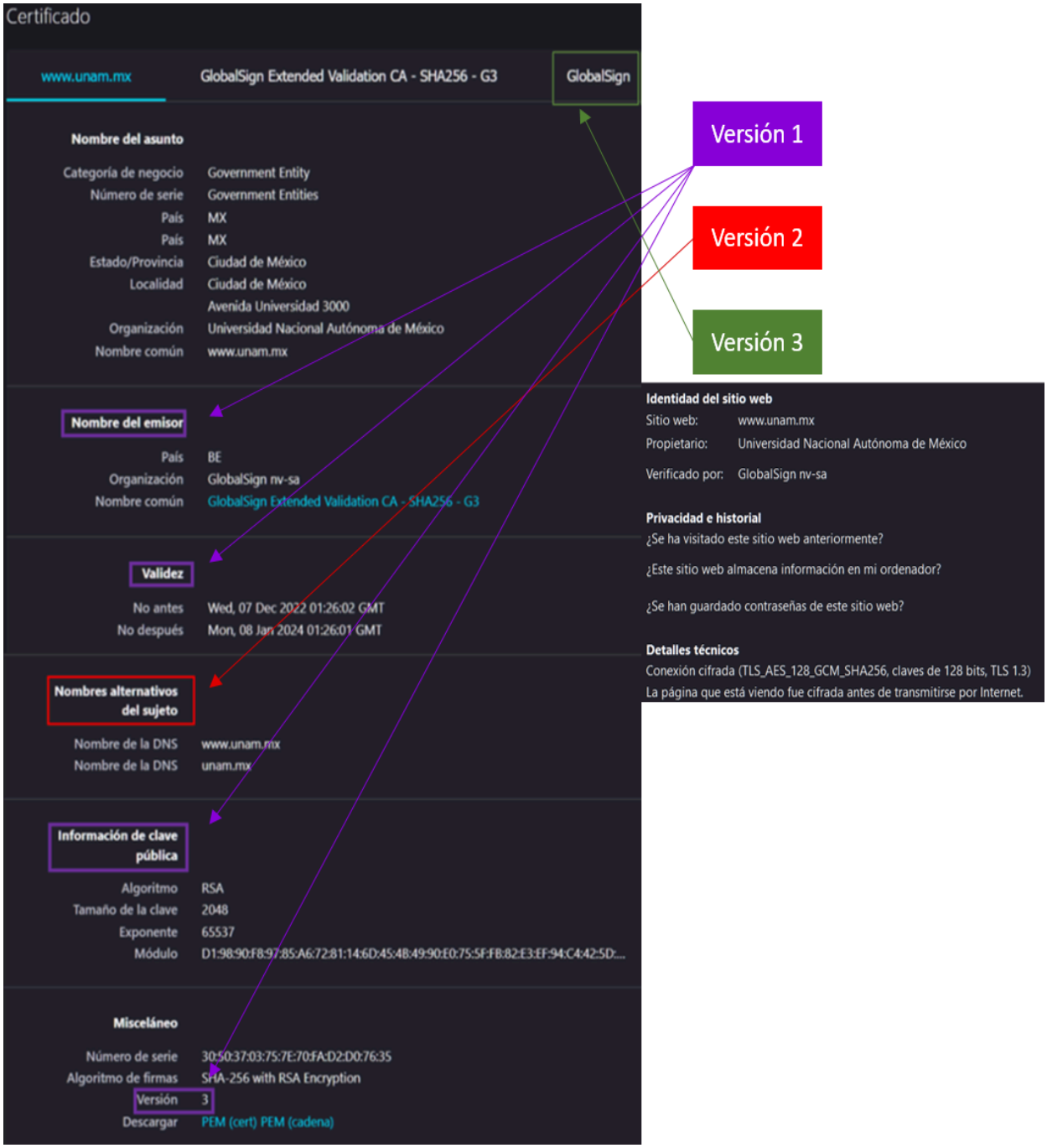

Imagen 8: Ejemplo de un certificado TLS v1.3. En la imagen hay recuadros de color morado que hacen referencia a campos de la versión 1, en color rojo podemos ver campos de la versión 2. Por último en color verde podemos ver campos de la versión 3, como la Autoridad certificadora que es quien emite el certificado. Para este ejemplo es GlobalSign.

Quien emite los certificados son las Autoridades Certificadoras (CA) ver imagen 9. Estos certificados se pueden comprar, adquirir a una entidad CA.

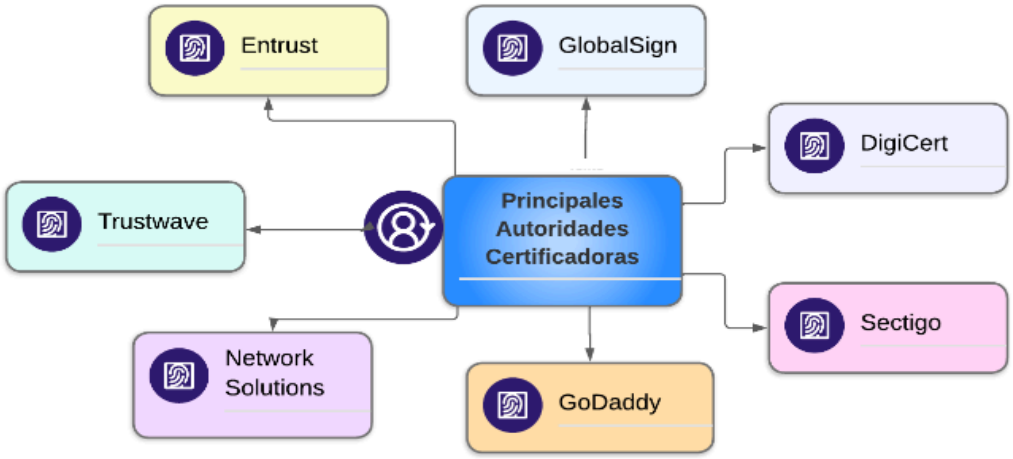

Imagen 9: Principales autoridades certificadoras.

3.2. JSON Web Token

Json web token por sus siglas JWT representa un estándar abierto el cual fue publicado en el RFC 7519. JWT define un método compacto y seguro para encapsular, compartir y transmitir aserciones (Claims) sobre una entidad (Subject) entre partes mediante el uso de objetos JSON [25].
El rasgo más importante de JWT es la estandarización en una forma simple, opcionalmente firmado y/o encriptada en un formato compacto.
El uso de JWT es muy común en aplicaciones web y servicios para autenticar usuarios y transmitir información de forma segura entre el cliente y el servidor. JWT compacto y autocontenido, es una opción popular para la autenticación y autorización en aplicaciones distribuidas y sistemas que utilizan arquitecturas de microservicios.

3.2.1. Historia de los JWT

En el año 2011 se formó el grupo "JOSE" por sus siglas en inglés JSON Object Signing and Encryption. Este grupo tuvo como objetivo estandarizar el mecanismo para la protección de integridad y encriptación. Así como el formato para claves y los identificadores de los algoritmos para apoyar la interoperabilidad de los servicios de seguridad para protocolos que utilizan JSON. [26]

Para el año 2013 ya se habían publicado una serie de borradores, incluido un libro de "recetas" con ejemplos de uso de las ideas producidas por el grupo. A partir de esos borradores surgen posteriormente los siguientes RFCs: JWT, JWS, JWE, JWK y JWA.

- JSON Web Signatures (**JWS**), RFC 7515.
- JSON Web Encryption (**JWE**), RFC 7516.
- JSON Web Key (**JWK**), RFC 7517.
- JSON Web Algorithms (**JWA**), RFC 7518.

Para el año 2014 el grupo JOSE recibe el premio: el "European Identity & Cloud Awards 2014" por la nueva especificación JWT/JOSE. [27] La cual sienta las bases de las comunicaciones seguras basadas en API y son esenciales para las futuras comunicaciones móviles seguras y la IoEE (Internet of Everything and Everyone).

Desde su lanzamiento JWT ha tenido un gran soporte, difusión, asimilación para la autenticación en aplicaciones móviles o web. Con muchas librerías para distintos lenguajes de programación (ver tabla 4).

Lenguaje de programación	librería
Scala	https://github.com/jwt-scala/jwt-scala
ruby	https://github.com/jwt/ruby-jwt
go	https://github.com/dgrijalva/jwt-go
.Net	https://github.com/jwt-dotnet/jwt
python	https://github.com/jpadilla/pyjwt
java	https://github.com/auth0/java-jwt
Perl	https://metacpan.org/pod/Crypt::JWT
Javascript	https://github.com/auth0/jwt-decode

Tabla 4: Lenguajes de programación con soporte JWT.

3.2.2. Estructura JWT

La representación de un JSON Web Token (JWT) está compuesta por una serie de caracteres alfanuméricos separados por un punto ver imagen 10.

"JWT es una forma compacta, imprimible y segura de representar datos cifrados mediante estructuras JSON"

Un JWT se compone de tres partes: Header (cabecera), Payload (carga útil) y signature (firma/cifrado).
La cabecera y la carga útil son objetos JSON estructurados, mientras que la firma/cifrado depende del algoritmo utilizado para firmar o cifrar el token. Si el token no se firma, la tercera parte se omite.
Cada parte se codifica de forma separada. Específicamente, la cabecera y la carga útil se codifican en Base64 y se concatenan con un punto. La firma se crea a partir de la concatenación de la cabecera y la carga útil, y luego se cifra según el algoritmo definido en las especificaciones JWS/JWE. El resultado final es un token compacto y seguro que se puede compartir fácilmente entre sistemas.

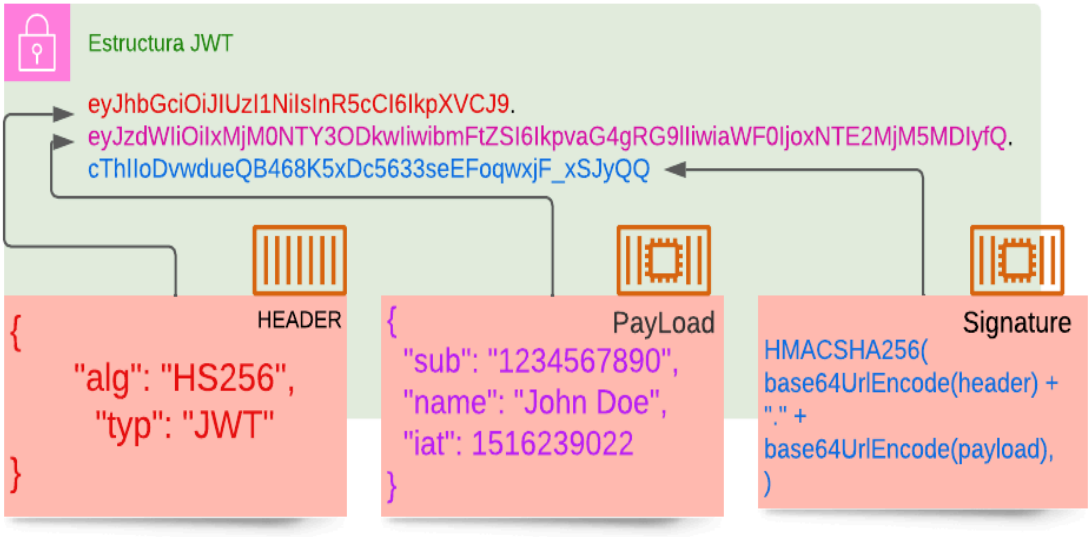

Imagen 10: Estructura de un JWT.

Es fundamental tener en cuenta que los JSON Web Tokens (JWT) no se encuentran cifrados[8], a menos que se utilice la especificación JWE (JSON Web Encryption). Con JWE, el contenido del payload es cifrado y posteriormente, firmado con JWS (JSON Web Signature). Para descifrar el contenido, se debe proporcionar una contraseña común o una clave privada. De esta manera, se verifica al remitente, se mantiene la confidencialidad y autenticidad del mensaje, y el payload no puede ser leído como texto plano tras el descifrado de Base64.

[8] El cifrado es una técnica empleada en informática para salvaguardar la privacidad y la seguridad de la información, asegurando que únicamente personas autorizadas puedan acceder y comprender los datos.

La cadena que se muestra en la imagen 10 es una serialización que utiliza la codificación base64url[9]. Esto permite que el contenido JSON del token pueda ser decodificado fácilmente para visualizarlo en texto plano imagen 11.

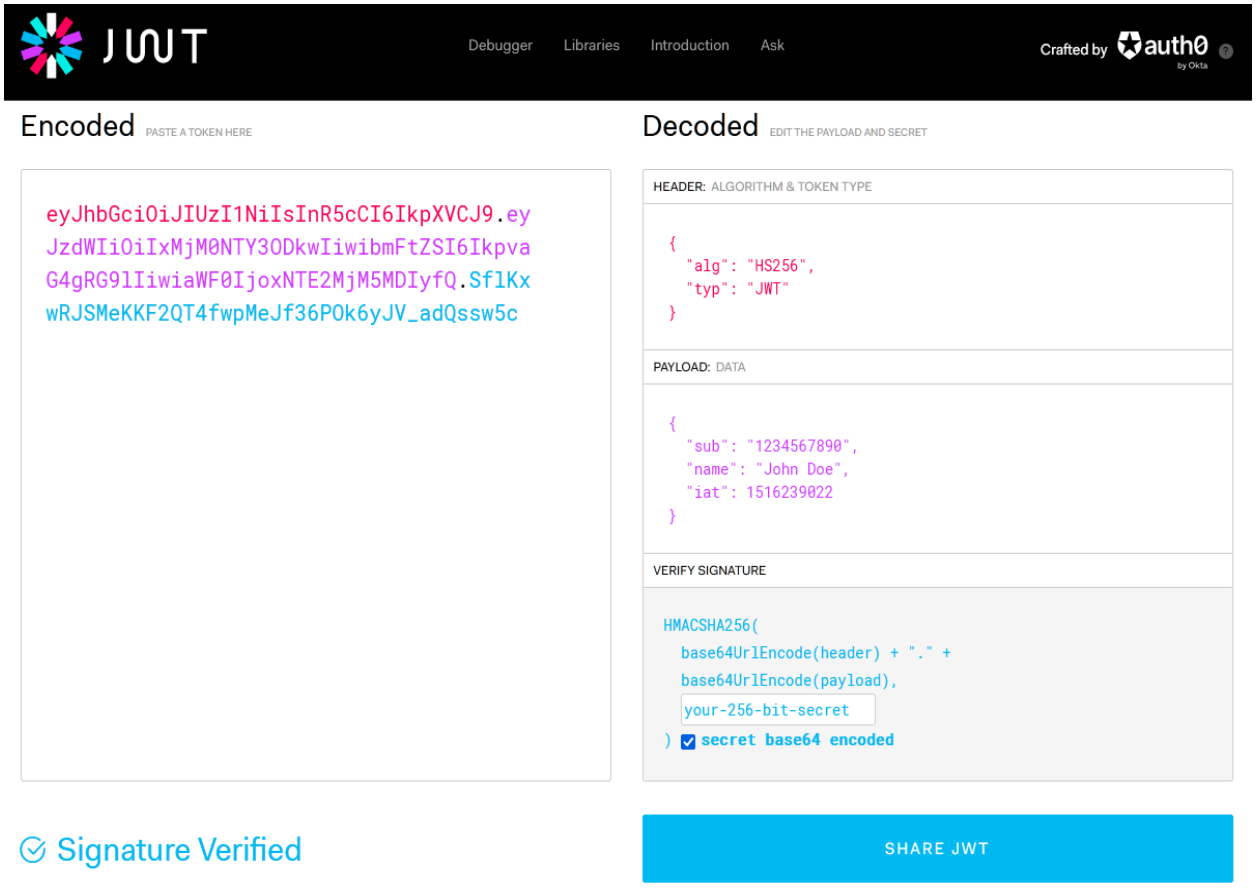

Imagen 11: Decodificación de un JWT. En la imagen podemos ver el contenido del JWT gracias a jwt.io que es una herramienta disponible en internet para verificar el contenido del JWT así como su firma.

El propósito principal del JWT no es encriptar los datos, sino proporcionar una forma segura de transmitir información entre diferentes sistemas. El JWT utiliza una firma digital para verificar que los datos no han sido manipulados durante la transmisión.

[9] Base64url es una forma de codificar datos binarios en texto ASCII, facilitando su transporte a través de la web y otros medios. Aunque su proceso de codificación es similar a la Base64 tradicional, emplea un conjunto de caracteres diferente y elimina algunos caracteres que pueden generar inconvenientes en ciertos contextos, como los signos de igual (=) y más (+).

3.2.2.1. Header

El header (encabezado) es la parte inicial del token que contiene información sobre el algoritmo de cifrado utilizado y el tipo de token. Describe las operaciones criptográficas aplicadas al JWT y propiedades adicionales del JWT como se observa en la Tabla 5.

Header		
Claim	**Nombre**	**Descripción**
typ	Token type	Tipo de token a usar JWT / (JWS,JWE).
cty	Content type	Debe estar presente en casos de firma, encriptado (JWS, JWE). Es utilizado para transmitir información estructural sobre el JWT.
alg	Mensaje de autentificación /algoritmo de firma.	Los JWT cifrados incluyen detalles sobre los algoritmos criptográficos empleados tanto cifrado principal como el cifrado del contenido. HEADER: `{ "alg": "ES256", "typ": "JWT" }` Tres algoritmos de firma para JWT. 1. Cifrado simétrico HMAC [código de verificación de mensaje hash]: HS256 / HS384 / HS512 2. Cifrado asimétrico RSASSA [algoritmo de firma RSA] (RS256 / RS384 / RS512) 3. ECDSA [Algoritmo de firma de datos de curva elíptica] (ES256 / ES384 / ES512)

Tabla 5: Campos del header en JWT.

3.2.2.2. Payload

El Payload (carga útil) del JWT contiene los "claims" o reclamaciones con información relevante, como el emisor del JWT, el usuario y el tiempo de expiración del token (Ver tabla 6). Podríamos definir al Payload como en donde se suelen añadir los datos del usuario.
Existen los siguientes dos tipos de Claims:

- Registered Claims: Son aquellos Claims ya definidos en la propia especificación JWT, muy parecidos a los campos usados en x.509.
- Public and Private Claims: Son todos aquellos Claims que no forman parte de los Registered Claims. Estos pueden ser Private Claims ó Public Claims.
    - Private Claims: Están definidas por usuarios (consumidores[10] y productores[11]) del JWTs. Es decir son utilizados para un caso en específico.
    - Public Claims: Son aquellos que se encuentran registrados en el registro de Claims de JSON Web Token que mantiene la IANA (Internet Assigned Numbers Authority).

PayLoad		
Claim	**Nombre**	**Descripción**
iss	Issuer	El IDP quién emitió el JWT. Puede ser una cadena de String o una URI (uniform resource identifier).
sub	Subject	Objeto/Usuario en nombre del cual se emitió JWT.
aud	Audience	Audiencia/Receptores a quienes va dirigido el token. Es donde se va a consumir el recurso.
exp	Expiration time	Tiempo de vencimiento del token, después de este tiempo no es aceptado. Definido en formato POSIX.
nbf	Not before	Tiempo antes del cual el token no debe ser aceptado. Este claim coloca el momento exacto desde el cual el JWT es considerado válido. Definido en formato POSIX.
iat	Issued at	Identifica el tiempo en el cual el JWT fue emitido.
jti	JWT ID	Identificador Único para el JWT.

Tabla 6: Tipos claims usados en el Payload de los JWT.

Se puede observar una similitud entre los claims de los JWT y los campos previamente vistos en los certificados X.509, tal y como se muestra en la tabla 7.

[10] Un consumidor es la parte que recibe los datos producidos y que puede validar o descifrar dichos datos.
[11] Un Productor es la parte que crea o genera los datos y aplica una firma o un cifrado a los mismos.

Claims	JWT	X.509
iss	En JWT, el "Issuer" (también conocido como "iss") indica la entidad que emitió el token.	En X.509, el campo "Issuer" indica la entidad que emitió el certificado.
sub	En JWT, el "Subject" (también conocido como "sub") indica el sujeto del token, es decir, el usuario o entidad a la que se refiere el token.	En X.509, el campo "Subject" indica el nombre del titular del certificado.
aud	En JWT, la “Audience” es a quienes va dirigido el token. Es donde se va a consumir el recurso.	En X.509, no hay un campo directamente equivalente al "Audience" (también conocido como "aud") de JWT. Sin embargo, los certificados pueden tener extensiones de "Nombre alternativo del sujeto" (SAN) que indican los nombres adicionales para los que se emitió el certificado.
exp	En JWT, el "Expiration time" (también conocido como "exp") muestra fecha y hora de expiración del token.	En X.509, el campo "Not After" muestra fecha y hora de expiración del certificado.
nbf	En JWT, el "Not before" (también conocido como "nbf") muestra fecha, hora en el cual el token es válido.	En X.509, el campo "Not Before" muestra fecha, hora desde la cual el certificado es válido.
iat	En JWT, el "Issued at" (también conocido como "iat") muestra fecha, hora cuando se emitió el token.	En X.509, el campo "Valid From" muestra fecha, hora cuando se emitió el certificado.
jti	Para JWT, es el Identificador Único para el JWT.	En X.509, no hay un campo directamente equivalente al "JWT ID" (también conocido como "jti") de JWT. Algunas implementaciones de X.509 pueden tener extensiones personalizadas que incluyan identificadores únicos para el certificado.

Tabla 7: Similitudes entre Claims y Campos X.509. Es importante tener en cuenta que estas son solo comparaciones generales y no hay una correspondencia exacta entre los campos de JWT y X.509.

3.2.2.3. No Signature

Se pueden crear JWT no seguros, los cuales omiten la parte signature (sin firmar). Estos tipos de JWT podrían ser útiles para ciertos casos de uso del lado del cliente, especialmente si los datos no son sensibles y sólo se usan para construir una vista para el usuario. Si un usuario malintencionado intenta cambiar estos datos, no obtendrá ningún beneficio.

A continuación se muestra el proceso de como crear un token inseguro/No signature ver imagen 12.

1. Se toma la parte del header como un arreglo de bytes representado en UTF-8.
2. Se codifica el arreglo usando el algoritmo base64-URL, removiendo signos de igual (=).
3. Se toma la parte del payload como un arreglo de bytes representado en UTF-8.
4. Se codifica el arreglo usando el algoritmo base64-URL, removiendo signos de igual (=).
5. Se concatena el resultado de los pasos anteriores. Se coloca primero el resultado del header (en color rojo) seguido por un punto " . " seguido por el resultado del payload en morado.

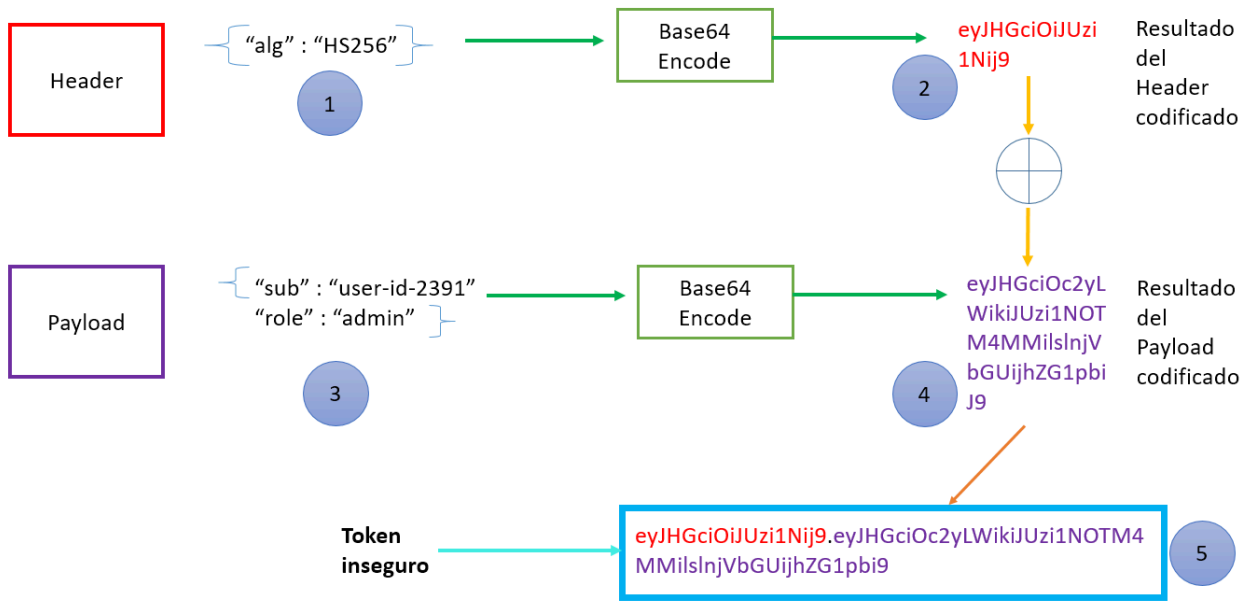

Imagen 12: Creación de un token inseguro.

3.2.2.4. Signature

Los JSON Web Signatures (con firma o firmados) son probablemente la característica más útil de los JWT. Al combinar un formato de datos con una serie de algoritmos de firma bien definidos. Esta es la razón por la cual JWT se está convirtiendo rápidamente en el formato para compartir datos de forma segura entre clientes e intermediarios.
El propósito de firmar es permitir a una o más partes establecer la autenticidad del JWT. En este contexto la autenticidad significa que los datos contenidos en el JWT no han sido manipulados.
La firma no impide que otras partes lean el contenido de JWT, el cifrado es quien cubre este rol de protección.

A continuación se muestra el siguiente Token JWT:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.
TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

En este token podemos observar las tres partes que componen un JWT: Header, Payload, Signature.
Hay varios algoritmos de firma disponibles según la especificación JWS.

- HMAC usando SHA-256, llamado HS256 en la especificación JWA.
- RSASSA PKCS1 v1.5 utilizando SHA-256, denominado RS256 en la especificación JWA.
- ECDSA utilizando P-256 y SHA-256, denominado ES256 en la especificación JWA.

JWA es la especificación JSON Web Algorithms, RFC 7518.
En la imagen 13 podemos ver el proceso de codificación de un JWT con firma. De manera similar a la imagen 12 de la sección anterior. [26]

Los pasos son los siguientes:

1. Se codifica el header usando el algoritmo base64-URL, removiendo signos de igual (=).
2. Se codifica el payload usando el algoritmo base64-URL, removiendo signos de igual (=).
3. Se concatena el resultado de los pasos anteriores. Se coloca primero el resultado del header (en color rojo) seguido por un punto “ . ” seguido por el resultado del payload en morado.
4. Se utiliza un secreto compartido (encriptar), un algoritmo de firma sobre el token sin firma.
5. Se firma el token y se codifica en Base64-URL.
6. Para obtener un token firmado, se concatena el token sin firma y la firma generada en el punto cinco.

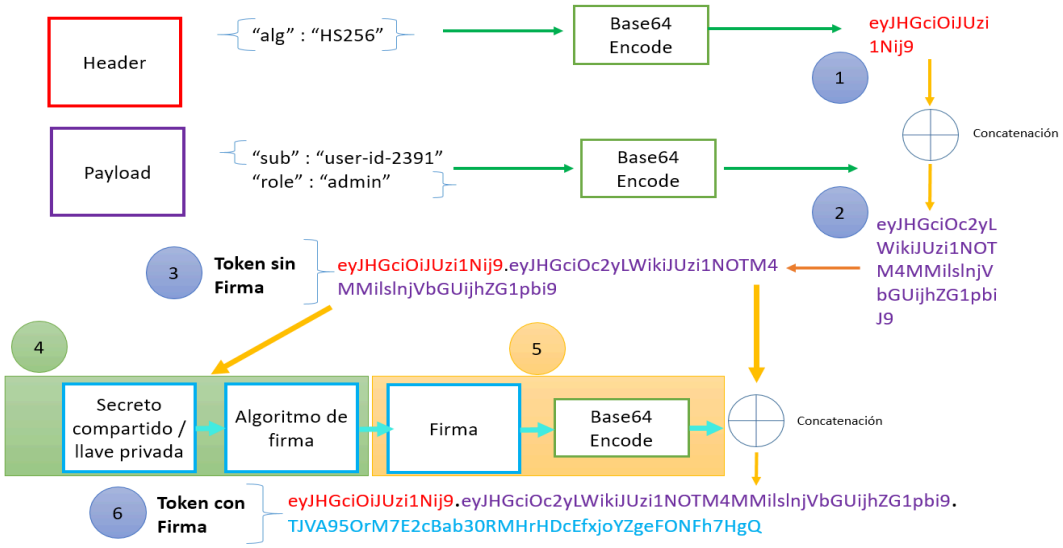

Imagen 13: Creación de un Token con firma.

JSON Web Signature (JWS) proporciona un medio para validar los datos, JSON Web Encryption (JWE) proporciona una forma de mantener los datos opacos (ilegibles) a terceros. Los tokens cifrados no pueden ser inspeccionados por terceros.

3.2.3. Esquemas JWS y JWE

Los estándares JWS, JWE proporcionan esencialmente dos esquemas: un esquema de secreto compartido y un esquema de clave pública/clave privada.

- **Esquema de Secreto Compartido: JWS**

  El esquema de secreto compartido funciona cuando todas las partes conocen el secreto compartido. Cada parte que posee el secreto compartido puede cifrar y descifrar información.
  Esto es análogo al caso de un secreto compartido en JWS: *las partes que poseen el secreto pueden tanto verificar cómo generar tokens firmados.*

- **El esquema de clave pública/privada: JWE**

  En JWE la parte que posee la clave privada es la única que puede descifrar el token. En otras palabras, quien tiene la llave pública[12] *puede cifrar datos, pero solo la parte que tiene la llave privada puede descifrar (y cifrar) esos datos.*

[12] "clave pública" o "llave pública" Ambos términos son utilizados de manera amplia y aceptada en el contexto de la criptografía asimétrica.

En JWE las partes que poseen la clave pública pueden introducir nuevos datos en un intercambio.

En JWS las partes que poseen la clave pública solo pueden verificar los datos pero no introducir datos nuevos.

JWS y JWE son complementarios cuando se usan en esquemas de clave pública/privada.
Para entenderlo mejor podemos verlo en términos de productor/consumidor (ver tabla 8).

- **Signature JWS**

El productor firma o cifra los datos para que los consumidores puedan validarlos o descifrarlos. **En el caso de las firmas JWT, la *clave privada* es utilizada para firmar (productor) JWTs, mientras que la *clave pública* se utiliza (consumidor) para validar.** El productor posee la clave privada y los consumidores la pública. El flujo de los datos sólo va a partir de quien tiene clave privada (productor) a quien tiene la clave pública (consumidor) ver imagen 14. [26]

- **Encryption JWE**

La clave pública se emplea para cifrar datos, mientras la clave privada para descifrar los datos. *En nuestro caso los datos sólo fluyen de quien tiene la clave pública (productores) a quien tiene la clave privada (consumidores)* ver imagen 15. [26]

	JWS	JWE
Productor	**Clave Privada**	**Clave Pública**
Consumidor	**Clave Pública**	**Clave Privada**

Tabla 8: Relación entre productores/consumidores en JWS y JWE.

Para ejemplificar estos dos conceptos de firma y encriptación en JWT se muestran dos diagramas que ejemplifican cada mecanismo.

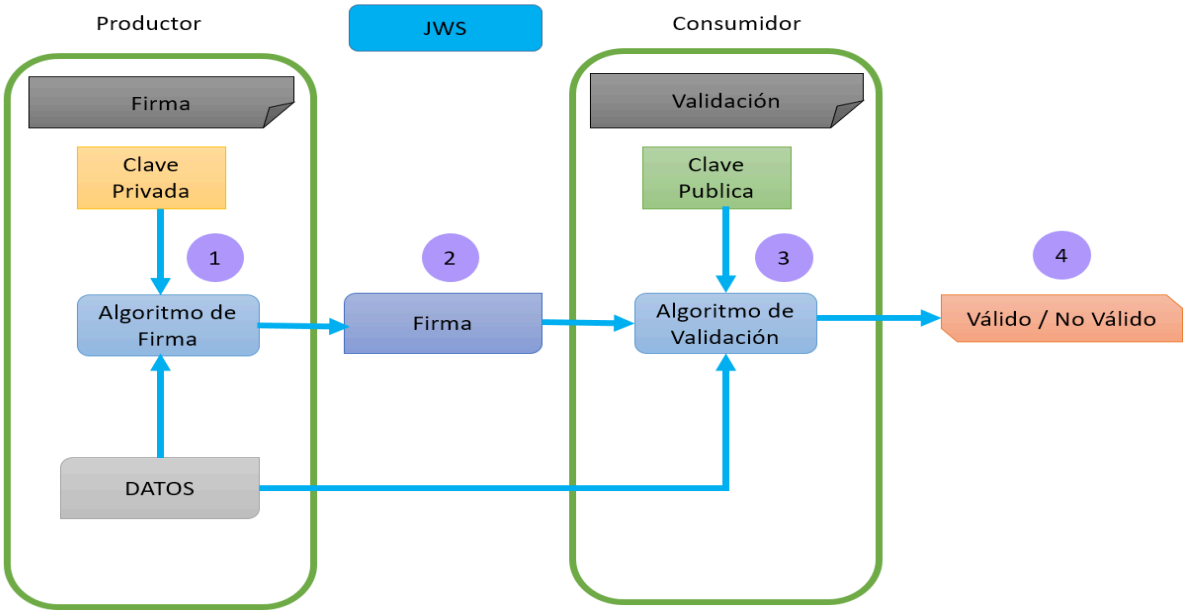

Imagen 14: Diagrama Productor/Consumidor JWS para firmar datos. El productor es quien tiene la clave privada y el consumidor quien tiene la clave pública. Los datos entran a un algoritmo de firma "productor" (1), después se obtiene la firma (2). El consumidor a partir del algoritmo de validación (3) y la clave pública se dictamina si es válido o no la firma (4). Aquí solo se garantiza la autenticidad.

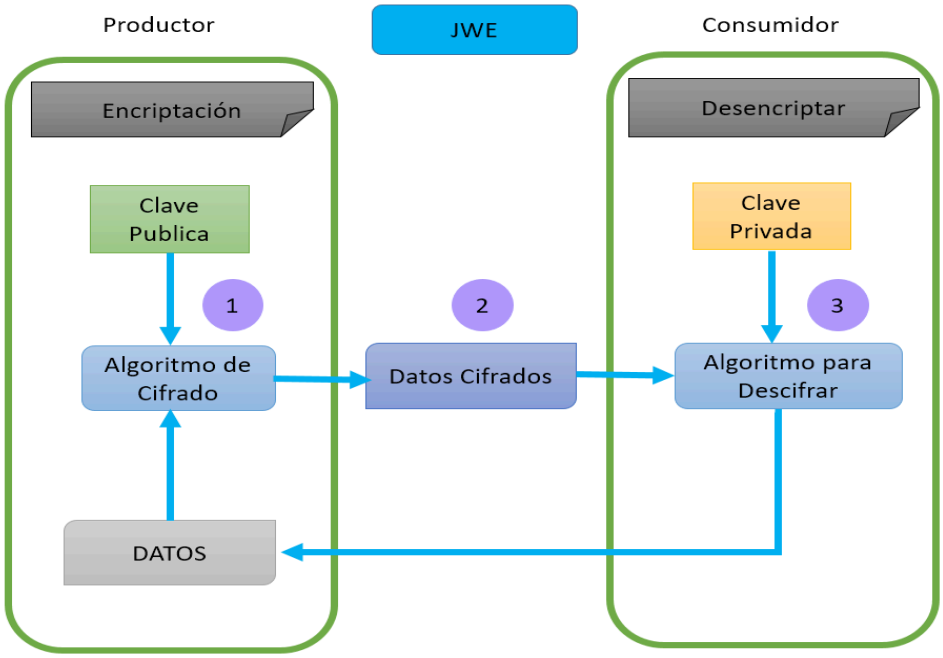

Imagen 15: Diagrama Productor/Consumidor JWE para cifrar/descifrar datos. Bajo el esquema de clave pública/privada. El productor se encarga de encriptar los datos usando la clave pública (1), los datos son encriptados (2). El consumidor con su clave privada y el algoritmo de encriptado puede acceder a los datos (3).

3.3. Cómputo de Alto Rendimiento

La computación de alto rendimiento es una tecnología que aprovecha la capacidad de supercomputadoras o clusters para abordar problemas complejos que demandan de un procesamiento intensivo.
Es aquí donde aparecen estas dos vertientes del cómputo de alto rendimiento HPC[13] y HTC[14].

- **HPC:** Se caracteriza por la ejecución de trabajos paralelos muy acoplados, que requieren un gran poder de cálculo intensivo. Este se ejecuta en un solo sitio con interconexiones de baja latencia.
- **HTC:** Permite un gran número de cálculos seriales o de memoria compartida. Es decir cálculos independientes o poco acoplados.

3.3.1. HPC

El Cómputo de alto rendimiento (High Performance Computing ó HPC) hace uso de computadoras potentes con alta capacidad en cómputo para resolver problemas cuya escala (en medida de datos o complejidad de operaciones) no hace factible su ejecución en un equipo convencional o de escritorio. El HPC no sólo se relaciona con los algoritmos (códigos) sino con el hardware, donde éstos deben correr en forma eficiente y certera" [28].

El cómputo de alto rendimiento HPC, Se centra en la capacidad de procesar datos, realizar cálculos complejos a velocidades muy altas. Uno de los tipos de soluciones HPC más conocidos y con lo cual estamos más familiarizados son las supercomputadoras. Los problemas que se abordan con HPC son problemas complejos, altamente acoplados, con gran carga computacional en los cuales se pretende utilizar ese poder de cómputo para resolverlos. Una de las maneras de aprovechar el poder de cómputo en HPC es el uso del cómputo paralelo [29].

3.3.2. HTC

El High Throughput Computing hace uso de muchos recursos informáticos durante largos periodos de tiempo para lograr completar un conjunto de tareas (ver tabla 9). The European Grid Infrastructure lo define como "Un paradigma de computación que se enfoca en la ejecución eficiente de gran número de tareas poco acopladas". Throughput es la medida de cuántas unidades de información puede procesar un sistema en una cantidad de tiempo determinada. Algunos de los problemas que se abordan son: Alineación de las secuencias de ARN/ADN, optimización de modelos estadísticos, barrido de parámetros, análisis de múltiples imágenes [30].

[13] HPC "High Performance Computing" Es la capacidad de procesamiento de computadoras y sistemas altamente especializados y optimizados para realizar cálculos complejos y exigentes en términos de recursos y tiempo.
[14] HTC "High Throughput Computing" Es un modelo de cómputo distribuido enfocado en utilizar al máximo los recursos de cómputo disponibles.

	Trabajos ideales	Trabajos aún factibles	Menos factible, pero posible
Cores	**1**	**< 8**	**> 8 ó MPI[15]**
(GPUs)	1 sin especificar tipo	1 tipo específico de GPU	Múltiples GPUs
Tiempo	< 12 Horas	< 24 Horas	> 24 Horas
RAM	< pocos GB	< 10 GB	> 10 GB
Input	< 500 MB	< 10 GB	> 10 GB
Output	< 1 GB	< 10 GB	> 10 GB
Software	Portable (precompilados binarios, contenedores)	Más que ->	Software con licencia: no Linux

Tabla 9: Trabajos más afines a HTC según OSG [31]. La tabla describe cómo ven los trabajos HTC en Open Science Pool. Estos deben de contar con una forma y tamaños particulares por trabajo para que puedan ejecutarse. De esta manera no están sujetos a tasas de interrupción ya que eso empeora el rendimiento de ese gran sistema distribuido.

[15] MPI "Message Passing Interface": Es una especificación de *programación* para la comunicación de procesos en sistemas de cómputo paralelo y distribuido.

3.3.3. Diferencias entre HPC y HTC

Como ya vimos, dentro del campo del cómputo de alto rendimiento, hay dos enfoques principales: High Performance Computing (HPC) y High Throughput Computing (HTC) ver tabla 10. Aunque ambos términos se refieren a la capacidad de procesamiento de grandes cantidades de datos, hay diferencias fundamentales en la forma en que se utilizan los recursos de computación.

HPC **High Performance Computing**	**HTC** **High Throughput Computing**
Las tareas de HPC se caracterizan por necesitar grandes cantidades de poder de cómputo por cortos periodos de tiempo.	Las tareas HTC requieren grandes cantidades de cómputo pero por tiempos largos. (meses, años, raramente horas o días).
Las tareas son medidas regularmente en floating point operations per second (FLOPS).	Las tareas son medidas regularmente por la cantidad de trabajos completados sobre un periodo de tiempo largo. (Mes, año).
Más enfocado en la rapidez al realizar tareas.	Enfocado más en la cantidad de trabajos completados en un largo tiempo.
HPC tiende a enfocarse en trabajos paralelos altamente acoplados que necesitan comunicación entre las tareas.	HTC trabajos generalmente secuenciales, independientes.
Generalmente implica ejecutar una sola instancia de software paralelo en muchos procesadores.	Implica ejecutar instancias independientes múltiples de software en múltiples procesadores al mismo tiempo.
Las tareas se comunican en mayor o menor medida para realizar los cálculos.	Tareas similares pero no altamente paralelas. Correr el mismo trabajo con diferentes variables de entrada.
Requiere programación paralela con MPI “Message Passing Interface”.	Paralelización Natural (muchos archivos o parámetros) para correr.
Requiere sistemas de cómputo especializados.	Requiere un sistema informático con componentes básicos.

Tabla 10: Comparativa HPC, HTC principales diferencias.

3.4. HTCondor

HTCondor es un software para la gestión de trabajos en cómputo distribuido es de código abierto y fue desarrollado en la Universidad de Wisconsin-Madison. Su nombre proviene de "High Throughput Computing Condor". Este software permite a los usuarios aprovechar los recursos de cómputo distribuido de manera eficiente y escalable.

3.4.1. Introducción

HTCondor, originalmente conocido como Condor, es un sistema de manejo de trabajos (batch system) desarrollado en la Universidad de Wisconsin-Madison. Su desarrollo comenzó en 1984 como un proyecto de investigación. La primera versión funcional de Cóndor fue lanzada en 1988.
Siendo su autor Miron Livny quien combina:

1. El tema de su tesis doctoral en procesamiento cooperativo
2. El proyecto Remote Unix[16]
3. El proyecto Crystal Multicomputer[17]
4. El algoritmo Up-Down[18]

El concepto nace a partir de la idea de la flexibilidad, escalabilidad y fiabilidad. Donde se pretende hacer uso de los equipos de cómputo que se tienen a mano para obtener poder de cómputo. En muchas ocasiones hay equipos se mantienen ociosos y su poder de cómputo no es utilizado. Muchos de estos equipos son heterogéneos: con diferente sistema operativo, hardware y aplicaciones. Interconectados por redes poco fiables, con diferentes configuraciones, y de diferentes propietarios, cada uno con sus propias políticas y requisitos. Es aquí donde la flexibilidad es la clave, ya que a partir de redes poco fiables y equipos heterogéneos se creó un sistema fiable capaz de aprovechar ese poder de cómputo muchas veces no utilizado. [32]

HTCondor es un sistema de software que se utiliza para crear un entorno de High-Throughput Computing (HTC), lo cual significa que está diseñado para gestionar y distribuir eficientemente grandes cantidades de trabajos de cálculo intensivo. El objetivo de un entorno informático de alto rendimiento es proporcionar grandes cantidades de potencia computacional *tolerante a las fallas* en periodos prolongados de tiempo para usar eficazmente los recursos disponibles en la red.
Condor usa eficientemente el poder de cómputo de máquinas conectadas a través de una red.

[16] Remote UNIX fue un proyecto desarrollado en la universidad wisconsin-Madison durante la década de 1980 por Michael J. Litzkow y otros investigadores. Fue un proyecto innovador que aprovechó la capacidad de procesamiento ociosa de los sistemas informáticos distribuidos para ejecutar tareas de cómputo en segundo plano.
[17] El proyecto Crystal es un multicomputador diseñado e implementado por David DeWitt, Raphael Finkel y Marvin Solomon en la década de 1980.
[18] El algoritmo Up-Down es un algoritmo utilizado en el contexto de la programación distribuida y la sincronización de procesos. Su objetivo principal es permitir la coordinación y sincronización de múltiples procesos en un entorno distribuido.

Estos equipos de cómputo pueden integrar un clúster único, un conjunto de clusters, recursos en la nube, recursos locales o una Grid internacional. El poder proviene de la capacidad de aprovechar eficazmente los recursos compartidos con propiedad distribuida. [33]

3.4.2. HTCondor aspectos principales: trabajos, roles y demonios

El elemento principal en HTcondor son los jobs (trabajos) que son tareas computacionales simples. Las Tres partes principales de un trabajo son: La entrada, el ejecutable y la salida.
Los estados principales de un trabajo son: “Idle”, “Running" (ver imagen 18) y “Completed” (ver imagen 19). [35]
Para ver si un trabajo está inactivo “Idle” se usa el comando condor_q (ver imagen 16) el cual muestra el listado de trabajos presentes y el estado de los mismos. [35]

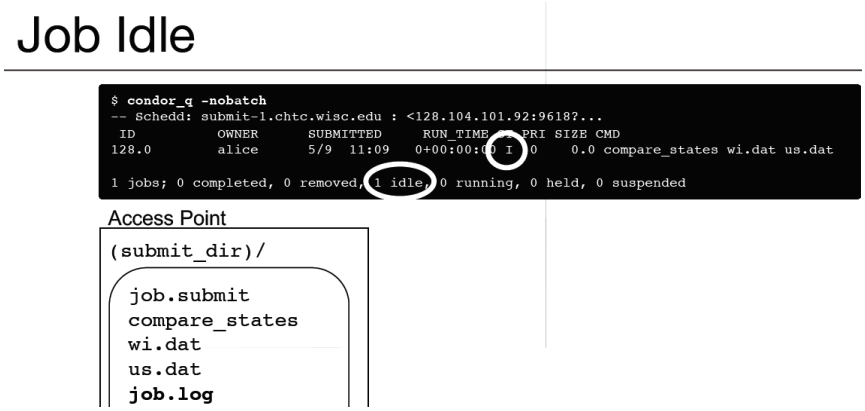

Imagen 16: Estado Idle de un trabajo en HTCondor.

Un trabajo se inicia cuando se manda desde el nodo submit al nodo execute. El archivo submit hace referencia a dos archivos de entrada wi.dat y us.dat para este ejemplo de job (ver imagen 17) .

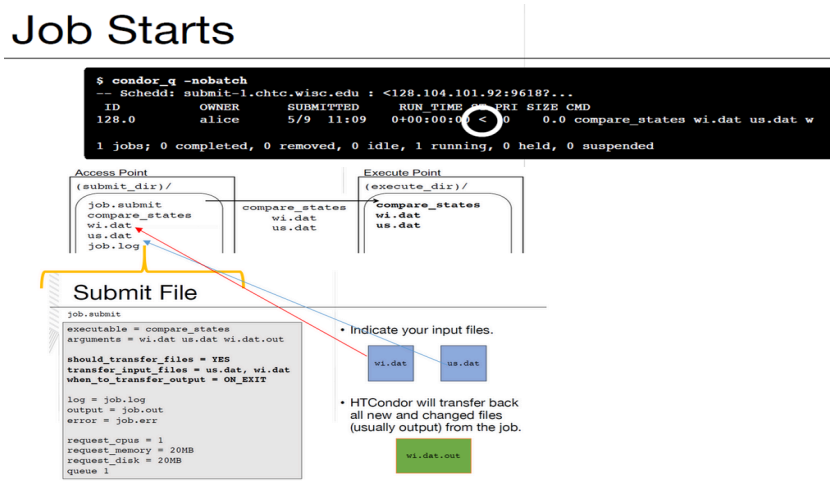

Imagen 17: Estado Starts de un trabajo en HTCondor.

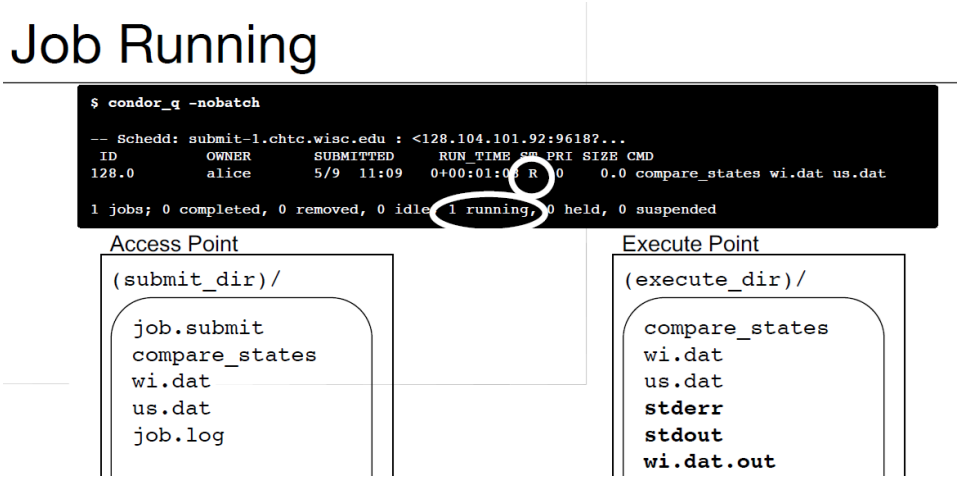

Imagen 18: Estado Running de un trabajo en HTCondor.

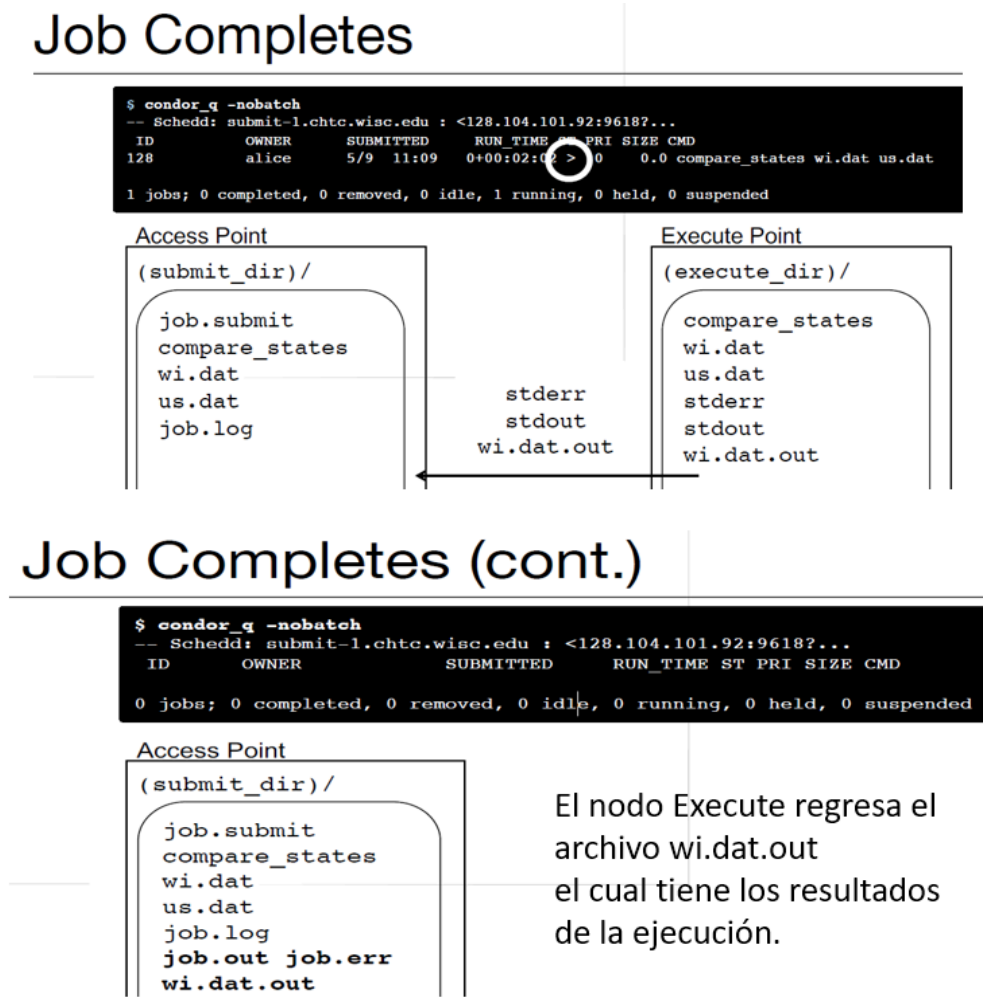

Imagen 19: Estado Completed de un trabajo en HTCondor.

Los usuarios envían sus trabajos de cálculo a HTCondor, el cual los encola y los ejecuta, proporcionando información sobre el resultado final del trabajo al usuario.
Los trabajos completados generan tres archivos los cuales contienen información importante referente al trabajo. Ver tabla 11.

Log	Output	Error
Cuando los trabajos son enviados, iniciados y detenidos. Se muestra: • Recursos utilizados • Estado de salida • Dónde se ejecutó el trabajo • Motivos de interrupción	Cualquier información que genere su programa como resultado.	Capturado por el sistema operativo.

Tabla 11: Archivos de salida de un trabajo HTCondor.

Se puede definir el universo para HTCondor como un entorno de ejecución para un trabajo. Este concepto de Pool o agrupación es importante ya que dentro del pool es donde se desarrollan los roles principales.
Dentro de HTCondor existen 3 roles principales: Execute, Submit y Central manager. ver imagen 20. [34]
Central Manager / Manejador central:

- Solo puede haber un administrador central para el grupo.
- Esta Máquina recolecta información.
- Se encarga de la negociación entre recursos y solicitudes de recursos.
- **Esta máquina es muy importante en el grupo HTCondor y debería ser confiable.**
- Si esta máquina falla no se pueden realizar más emparejamientos, aunque los emparejamientos actuales continúan hasta que alguna de las partes del emparejamiento rompa el emparejamiento.
- **Esta máquina 24/7 debe estar siempre disponible y que reinicie rápidamente.**

Execute/ Ejecutar:

- Cualquier máquina en el grupo, incluye al administrador central.
- Puede o no ejecutar trabajos de HTCondor.
- Estas máquinas no requieren de muchos recursos (salvo el disco duro).
- A mayor número de recursos en una máquina (espacio de intercambio, memoria, número de CPU) mayor variedad de solicitudes de recursos puede atender.

Submit /Enviar:

- Cualquier máquina en el grupo, incluye al administrador central.
- Puede o no permitir el envió de trabajos de HTCondor
- Una máquina de envío (recursos) >> Una máquina de ejecución (Recursos).

Central Manager: Administra y coordina las tareas que van a ser ejecutadas en el cluster.

Submit Node: Nodo de envíos de trabajos HTCondor.

Worker Node: Nodo capaz de recibir y ejecutar las tareas del nodo submit.

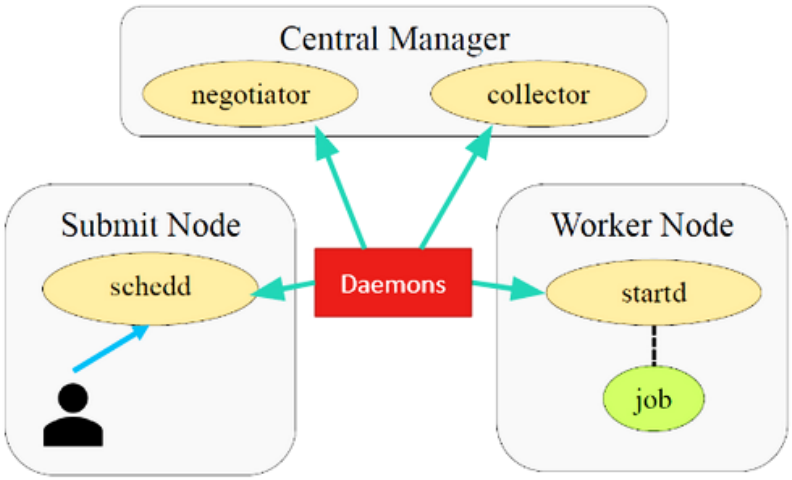

Imagen 20: Principales roles en HTCondor y su relación con los Demonios.

3.4.2.1. Demonios HTCondor

HTCondor utiliza demonios que cumplen diversas funciones, los mismos tienen diversos permisos, privilegios, tareas. A continuación se muestran solo *algunos* de los demonios de los más importantes:

condor_master:

- Es el demonio responsable de mantener al resto de demonios de HTCondor los cuales se ejecutan en cada máquina del grupo.
- Puede generar otros demonios.
- Si Existen nuevos archivos condor_master reinicia los demonios afectados.
- Si falla algún demonio, condor_master envía un correo al admin HTCondor del grupo y reinicia el demonios.
- Permite por medio de comandos: iniciar, detener o reconfigurar demonios remotamente.
- Se ejecuta en todas las máquinas del grupo.

condor_startd

- Se ejecutará en cualquier máquina del grupo que pueda ejecutar trabajos.
- Responsable de hacer cumplir la política que configura el propietario del recurso (condiciones que iniciar, suspender, reanudar anulan o eliminan trabajos)
- condor_startd listo para ejecutar un trabajo de HTCondor genera condor_starter.

condor_starter

- Genera el trabajo HTCondor en una máquina determinada.
- Configura el entorno de ejecución y supervisa el trabajo cuando se está ejecutando.
- Regresa información de estatus al completar un trabajo.

condor_schedd

- Se encarga de las solicitudes de recursos al grupo HTCondor
- Las submit machine deben correr este demonio
- Los trabajos van a condor_schedd, donde se almacenan en la cola de trabajos
- Condor_shchedd gestiona la cola de trabajos.
- Herramientas para cola de trabajos conectadas a condor schedd
    - *condor_submit*
    - *condor_q*
    - *condor_rm*
- Este demonio es responsable ya que reclama recursos disponibles. Para así atender las solicitudes.
- Genera un demonio condor_shadow cuando un trabajo se ha asociado con un recurso determinado.

condor_shadow

- Se ejecuta en la máquina donde se envió una solicitud determinada y actúa como administrador de recursos para la solicitud.

condor_collector

- Este demonio se encarga de recopilar toda la información relacionada con el estado de un grupo de HTCondor.
- Por lo general los demonios envían de forma periódica las actualizaciones del ClassAd (ClassAd contiene información acerca del estado de los demonios, recursos o solicitudes de recursos) a condor_collector.
- Condor_status comando para consultar al condor_collector sobre información específica sobre varias partes del HTCondor.

condor_negociador

- Es responsable de todas las coincidencias dentro del sistema HTCondor.
- Hace un ciclo donde consulta a condor_collector el estado actual de todos los recursos existentes en el grupo. Se comunica con el demonio condor_schedd “cola de trabajos” e intenta hacer coincidir los recursos disponibles con las solicitudes en la cola de trabajos.
- Es responsable de hacer cumplir con las prioridades de los usuarios en el sistema. Cuando un usuario demanda mayor recursos, menos prioridad tendrá para adquirir más recursos. 0.5 es la mejor prioridad (el valor más bajo), a mayor número será peor es la prioridad.

condor_kbdd

- Se usa en las plataformas Linux y Windows para determinar la actividad de consola (mouse o teclado). condor_kbdd se conecta al servidor X y verifica periódicamente si ha habido alguna actividad y si es así envía un comando a condor_startd.

condor_gridmanager

- Maneja la gestión y ejecución de todos los trabajos del universo grid. condor_schedd invoca el condor_gridmanager cuando hay trabajos del universo grid en la cola y condor_gridmanager sale cuando no hay más trabajos del universo en la cola grid.

condor_had

- Implementa la alta disponibilidad del administrador central de un grupo a través del monitoreo de la comunicación de los demonios necesarios. Si la máquina administradora actual deja de funcionar, este demonio asegura que otra máquina ocupe su lugar y se convierta en el administrador central del grupo.

condor_replication

- Ayuda a condor_had manteniendo una copia actualizada del estado del grupo.

condor_transferer

- Este demonio de corta duración es invocado por el demonio *condor_replication* para realizar la tarea de transferir un archivo de estado antes de salir.

condor_procd

- Este demonio controla y monitorea familias de procesos dentro de HTCondor.

condor_job_manager

- Transforma los trabajos del universo vainilla en trabajos del universo grid, para que los otros trabajos transformados puedan ejecutarse en otro lugar, según corresponda.

3.4.3. HTCondor Seguridad

Es muy fácil configurar una seguridad deficiente, por el contrario es difícil encontrar buenos consejos sobre seguridad robusta. Por otra parte, la seguridad puede resultar un tema complejo debido al gran espectro que abarca y la profundidad que se desee.
En HTCondor se presenta un flujo de seguridad el cual nos permite ver como se establece la interacción referente a la seguridad entre los demonios.

Flujo de seguridad general

1. **Negociación:** para la comunicación cliente-servidor se debe acordar qué mecanismos de seguridad se utilizan para la conexión. Las decisiones que toma el servidor durante la negociación se llaman “políticas de seguridad”.

2. **Autenticación / Asignación:** si el servidor decide autenticarse, se prueban los métodos permitidos en el orden decidido por el servidor hasta que uno de ellos tenga éxito. Después de una

autenticación exitosa, el servidor decide el nombre canónico del usuario en función de las credenciales utilizadas por el cliente.

3. **Cifrado e integridad:** Si el servidor decidió que se usaría el cifrado, ahora ambos lados harán las comprobaciones de cifrado e integridad utilizando el método preferido por el servidor. AES (Advanced Encryption Standard) es el método preferido y habilitado de forma predeterminada.

4. **Autorización:** El usuario se verifica para ver si puede enviar comando al servidor que desea enviar. Los comandos se "registran" en diferentes niveles de autorización y hay una lista de Allow/Deny para cada nivel. Si el usuario está autorizado HTCondor realiza la acción solicitada. Si falla la autorización, se niega el permiso y ***se cierra la conexión de red.***

A continuación se definirán unos conceptos importantes referentes a la seguridad en HTCondor:

- **Autentificación**: La identificación apropiada del usuario se logra por medio de la autentificación. Se discierne entre usuarios reales e impostores. De forma **determinada HTCondor usa la identificación del usuario UID para determinar la identidad.** Se cuenta con variedad de mecanismos incluidos kerberos y SSL.

- **Autorización**: ***Especifica quién puede hacer que***. HTCondor proporciona autorización por usuario o por máquina.

- **Privacidad**: HTCondor puede cifrar los datos enviados por la red. También puede cifrar los datos enviados para la comunicación interna así como datos del usuario, archivos y ejecutables. El cifrado opera en la transmisión de red, **los datos sin cifrar se almacenan en el disco de forma predeterminada.**

- **Integridad**: Una actividad que realiza HTCondor para verificar la integridad es enviar datos criptográficos adicionales y así corrobora que la transmisión de datos en red no fue alterada (El ataque del hombre en medio altera los datos, sin el conocimiento de ninguno de los extremos de la comunicación). La integridad se restringe a los datos enviados por la red, no siendo de la misma manera para los datos almacenados en disco (integridad de la información).

Riesgos de seguridad en HTCondor

1. HTCondor **No** puede evitar las brechas de seguridad de los usuarios que puedan elevar sus privilegios a la cuenta de administrador o root.

2. HTCondor **no** ejecuta trabajos de usuario en entornos limitados (excepto trabajos Dockers o Singularity).

3. **No** puede evitar un ataque de denegación de servicio DOS “Denial of Service”. Porque HTCondor asume que los usuarios son confiables.

4. HTCondor puede prevenir los accesos no autorizados al grupo HTCondor. En donde solo usuarios de confianza tienen acceso al grupo.

5. HTCondor provee de encriptación y verificación de integridad para garantizar que la transmisión en red no son examinadas o manipuladas mientras están en tránsito.

3.4.3.1. Métodos de Autenticación HTCondor

En HTCondor en el aspecto de la autenticación se cuenta con macros para especificar el tipo de autentificación. Este se divide en dos partes: la parte del cliente y del servidor. Estas macros se definen mediante una lista delimitada por comas o espacios de los posibles métodos a utilizar. La sección Autenticación enumera todos los métodos de autenticación implementados. A través de macros de configuración, tanto el cliente como el demonio puede especificar si se requiere cifrado para una comunicación posterior. El cliente usa una de dos macros para habilitar o deshabilitar el cifrado. Para el daemon, hay siete macros para habilitar o deshabilitar el cifrado (ver imagen 21).

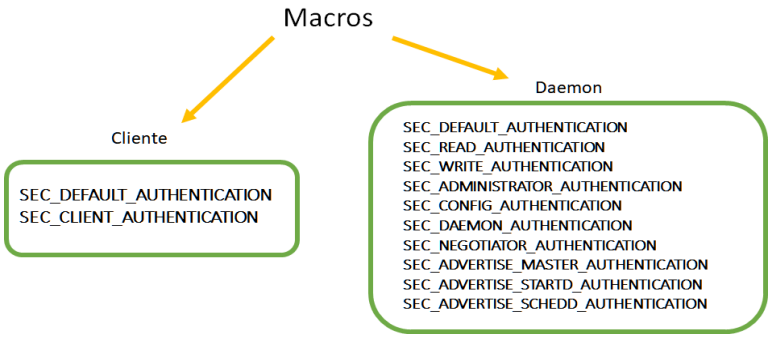

Imagen 21: Macros que utiliza HTCondor para cliente y Demonios.

Cuando se define una macro específica, el valor más específico tiene prioridad sobre la definición por defecto. El cliente puede proporcionar una lista de métodos aceptables, utilizando las macros. [36]
Si se va a realizar el cifrado, las partes que se comunican deben encontrar (negociar) un método de cifrado aceptable para ambas partes. El cliente puede proporcionar una lista de métodos aceptables, utilizando las macros (ver Imagen 22). El daemon puede proporcionar una lista de métodos aceptables, utilizando las macros.

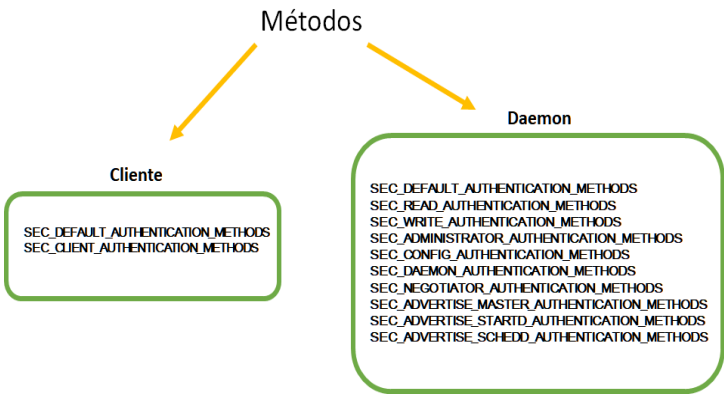

Imagen 22: Métodos que utiliza HTCondor para cliente y Demonios.

No todos los métodos de autenticación funcionan en plataformas Windows. [36] HTCondor cuenta con varios métodos de autenticación (ver tabla 12).

Métodos de Autentificación	**Consideraciones**
SSL	• Demonios y usuarios cuentan con certificados X.509
KERBEROS	• Su configuración es compleja. • Es de los métodos más seguros junto a SSL.
SCITOKENS	• El proyecto SciTokens pretende construir un ecosistema federado para la autorización en infraestructuras informáticas científicas distribuidas. Es usado para Tokens del tipo WLCG.
PASSWORD	• Para la comunicación entre demonios. No para autentificar usuarios finales solo para demonios. • Compatible con Windows, Unix.
FS	• No está disponible en plataformas Windows • Solo trabaja en máquinas locales.
FS_REMOTE	• No está disponible en plataformas Windows. • El Directorio destino debe estar correctamente configurado.
ID TOKENS	• Contiene una identidad. • Delimita las acciones permitidas (privilegios). • Requiere cliente y servidor.
NTSSPI	• Solo trabaja en Windows
MUNGE	• Servicio de autentificación y validación de credenciales. • altamente escalable. – Instalación de terceros.
CLAIMTOBE	• Muy inseguro. • Incluido en HTCondor Solo con Fines de prueba.
ANONYMOUS	• Es un método de autenticación más el cual tiene solo fines de prueba. No requiere configuración

Tabla 12: Métodos de autenticación que están disponibles en HTCondor.

3.5. HTCondor-CE

HTCondor-CE (2012) es un software basado en HTCondor configurado como un *Compute EntryPoint* (Punto de entrada) para sitios que forman parte de una red de cómputo mayor como una Grid.
La principal función es que sirve como puerta de entrada para solicitudes entrantes para la asignación de los recursos (RARs) "Resource Allocation Requests".
HTCondor-CE se encarga de la autorización y delegación de estas solicitudes al sistema batch local de un sitio Grid. Proporciona autenticación y autorización de clientes remotos. Los sistemas batch soportados según la documentación son: Grid Engine, HTCondor, LSF, PBS Pro/Torque y Slurm.

Podemos ver al punto de entrada "Compute EntryPoint" como una capa delgada de software instalado en un Host que funge la función de enviar y administrar trabajos en su sistema de lotes local. Ver imagen 23.

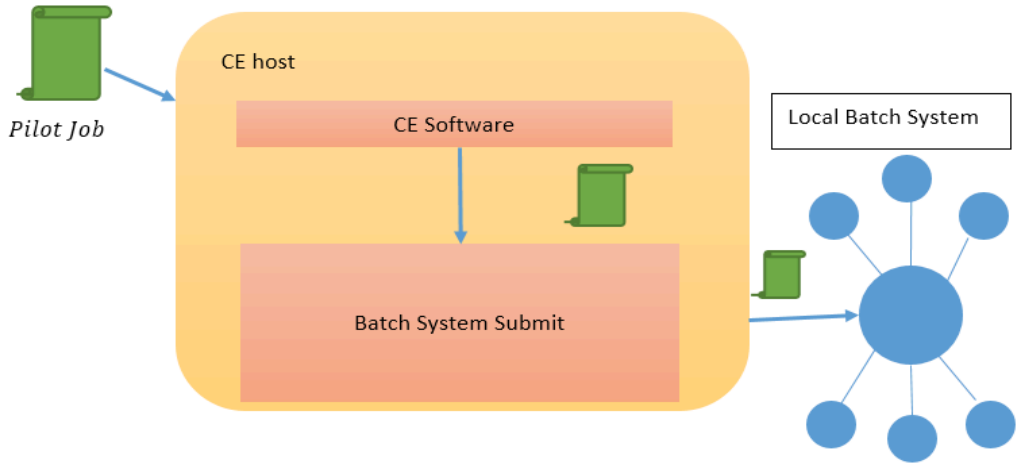

Imagen 23: Arquitectura "Compute Element". El Pilot Job se manda al **CE host** donde está instalado el CE software donde el Job Router crea una copia transformada (trabajo enrutado) se envía la copia al sistema Batch local. Después HTCondor-CE monitorea el trabajo en el sistema de lotes y comunica su estado al trabajo piloto original (Pilot Job), este a su vez notifica al remitente original de cualquier actualización.

Dentro de los beneficios de HTCondor-CE contamos con escalabilidad ya que soporta 16k RARs (Resource Allocations requests) concurrentes. También nos brinda herramientas de depuración, configuración de enrutamiento.

Una de las partes importantes de HTCondor-CE es el *Job Router Daemon*. Este demonio cambia de un tipo de trabajo a otro tipo de trabajo. Job Router es responsable de tomar un trabajo, crear una copia y cambiar la copia de acuerdo a un conjunto de reglas (enrutamiento de trabajos).

- Cada cadena de reglas se denomina "ruta de trabajo" y está definida por una classAd.
- Los jobs routes reflejan la política de los sitios.

Dentro de los Demonios principales en HTCondor-CE tenemos los siguientes, ver imagen 24.

- **Master**: Responsable de iniciar / parar otros demonios HTCondor en el host
- **SchedD**: acepta trabajos y almacena información sobre el estado de trabajo (por ejemplo: cola de trabajos).
- **Collector**: almacena la información sobre otros demonios HTCondor.
- **GridManager**: Envía trabajos a schedDs remotos en sistemas Batch no HTCondor.

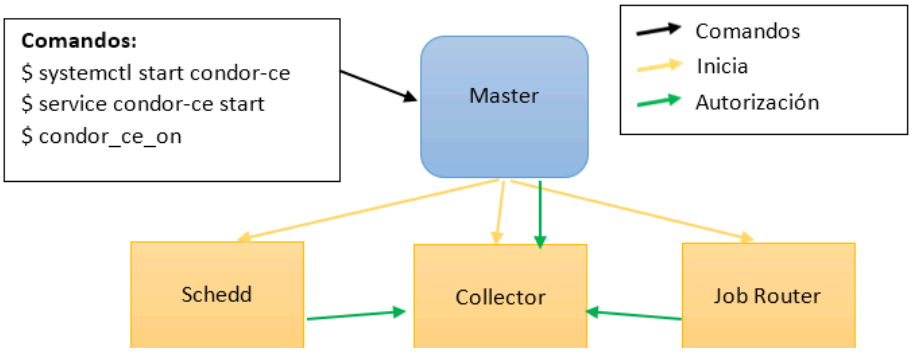

Imagen 24: Principales demonios HTCondor-CE y su interacción.

3.5.1. Estructuras HTCondor-CE

Dentro de la arquitectura HTCondor-CE existen varias configuraciones que usan diferentes sistemas batch. Estas configuraciones que aparecen dentro de la documentación HTCondor-CE son [38]:

- HTCondor-CE + HTCondor Batch System
- HTCondor-CE + Non-HTCondor Batch System
- HTCondor-CE + HTCondor + Non-HTCondor
- HTCondor-CE + SSH

En nuestra maqueta la arquitectura a usar es: HTCondor-CE + Non-HTCondor Batch System ver imagen 25.

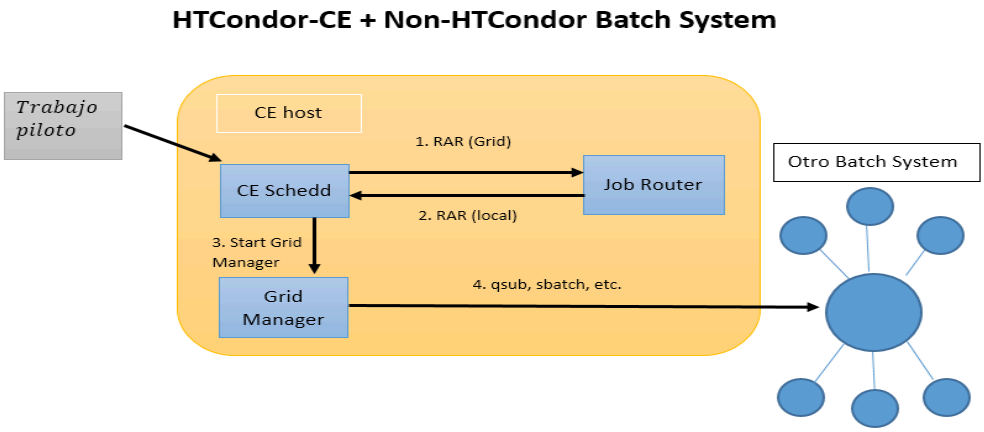

Imagen 25: Configuración HTCondor-CE + Non-HTCondor Batch System.

Para los sistemas no-batch HTCondor el Job Router transforma el trabajo piloto en un trabajo enrutado en el CE. El trabajo enrutado envía un trabajo al sistema batch a través de un proceso llamado BLAHP.
Con los sistemas por lotes que no son de HTCondor, HTCondor-CE no puede usar los protocolos internos de HTCondor para transferir archivos, por lo que su directorio "spool" debe exportarse un sistema de archivos compartido el cual se monta sobre los nodos trabajadores del sistema por lotes.

Consideraciones Previas antes de instalar HTCondor-CE [37]:

1. **ID de usuario**: Si aún no existen, la instalación creará el usuario condor de Linux (UID 4716).
2. **Certificado SSL**: El servicio HTCondor-CE utiliza un certificado de host y una clave para la autenticación SSL y GSI.
3. **Entradas de DNS**: El DNS de avance y retroceso debe resolverse para el host HTCondor-CE
4. **Puertos de red:** Las fábricas piloto deben poder contactar con su servicio HTCondor-CE en el puerto 9619 (TCP).
5. **Submit host**: HTCondor-CE debe instalarse en un host que tenga la capacidad de enviar trabajos a su clúster local, que soporte sistemas por lotes compatible (Grid Engine, HTCondor, LSF, PBS/Torque, Slurm).
6. **File System:** Los sistemas por lotes que no son HTCondor requieren un sistema de archivos compartido entre el nodo host HTCondor-CE y los nodos de trabajo.

3.6. Simple Linux Utility for Resource Management

En los sistemas HPC, HTC cuentan con muchos nodos y muchos usuarios. Se necesita de una herramienta para distribuir las tareas computacionales entre los nodos disponibles. Para esta función se emplea un sistema de lotes “Batch System”.

Slurm es un “Batch system” de código abierto. Inició en 2002 como un recurso manejador para cluster linux. Cerca de 500 mil líneas de código C. Soporta AIX, Linux, Solaris, otras variantes de UNIX.
Slurm gestiona clústeres y programación de trabajos. Este sistema robusto tolera fallas y es altamente escalable para clusters con linux.
Slurm como un administrador de carga de trabajos en un cluster ejerce tres funciones principales:

1. Slurm gestiona el acceso exclusivo o compartido de los nodos de cómputo para los usuarios durante el tiempo necesario. Permitiendo así la ejecución de trabajos.
2. Permite iniciar, ejecutar y monitorear las tareas.
3. Administra la disputa por los recursos gestionando la cola de trabajos pendientes, garantizando una eficiencia de los recursos disponibles.

3.6.1. Demonios SLURM

Slurm tiene un control central **slurmctld** el cual monitorea el estado de los recursos y el trabajo (Manejo de colas y asignación de recursos). Por lo general, existe un Slurmctld por clúster (Nodo de administración).
Cada Nodo de cómputo tiene un demonio Slurmd, parecido a un shell remoto que:

1. Espera trabajo
2. Ejecuta trabajo
3. Devuelve Estado

Los Demonios Slurmd proporcionan ciertas comunicaciones jerárquicas tolerantes a fallas. A continuación se describen los principales demonios en slurm:

- **slurmbd** (slurm database): opcional, puede usarse para registrar información contable para múltiples grupos administrados por slurm en una BD.
- **slurmrestd** (slurm rest api): este demonio se puede usar para interactuar con slurm a través de su API REST.
- **slurmctld:** Demonio de gestión central de Slurm.
- **slurmd:** Demonio de nodo de cálculo para Slurm.
- **slurmstepd**: Demonio manejador de pasos de trabajo para Slurm.

Algunos Comandos de Slurm:

- **srun**: para iniciar trabajos.
- **scancel**: para terminar trabajos en cola o ejecución.
- **sinfo:** reportar el estatus del sistema.
- **squeue**: para reportar el estatus de los trabajos.
- **sacct:** para obtener información sobre trabajos y pasos de trabajo que se están corriendo o se completaron.
- **sview**: informan gráficamente el sistema y el estado del trabajo, así como la topografía de red.
- **scontrol**: es una herramienta importante que permite monitorear, modificar tanto la configuración como la información del cluster.
- **sacctmgr**: Esta herramienta permite gestionar la base de datos del cluster.

Existe una relación entre los demonios del nodo maestro, los demonios en los nodos de cómputo y los comandos de slurm ver imagen 26. [39]

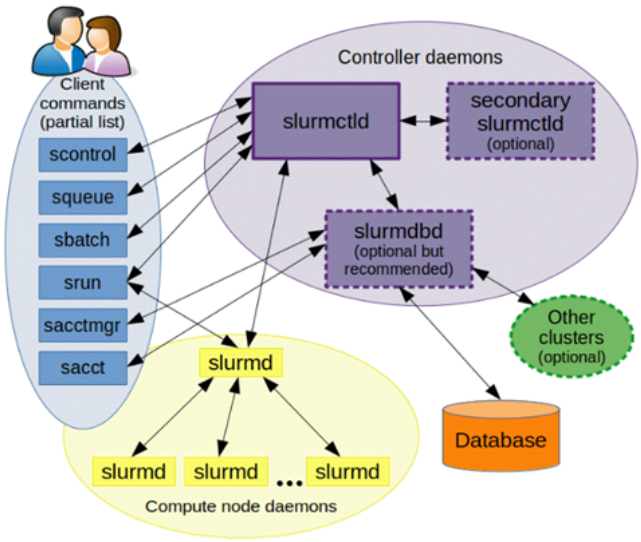

Imagen 26: Representación gráfica de los comandos SLURM y los demonios controladores del nodo maestro y los nodos de cómputo.

3.7. Autenticación y Autorización

La autenticación y la autorización son dos conceptos fundamentales en la seguridad informática. Estos conceptos se relacionan estrechamente con el control de acceso a sistemas y recursos.

<La **autenticación** es el proceso de verificar que la identidad de un usuario que solicita acceso a un sistema es auténtica. Este proceso implica la presentación de credenciales para asegurar que quien solicita el acceso es quien dice ser. La **autorización** se refiere al proceso donde se determina los derechos y permisos que tiene un usuario una vez que ha sido autenticado. No todos los usuarios tienen el mismo nivel de acceso a los recursos o servicios. *La autenticación puede llevar a la autorización pero al revés no, tener autorización no significa estar autenticado o identificado.*

Bajo este contexto una **entidad** es aquella persona, dispositivo o sistema capaz de interactuar con un sistema o recurso específico. Una entidad cuenta con **identidad** que es un conjunto de atributos que se pueden utilizar para distinguir esta entidad dentro de un contexto. Estos atributos pueden incluir un nombre de usuario, un número de identificación único, un certificado digital u otra información relevante. Por esto último una entidad puede ser identificada y autentificada.

Para ejemplificar los conceptos ver imagen 27. La imagen nos muestra como una entidad puede tener varias identidades, las cuales identifican a la entidad en dos contextos diferentes y que el observador ve a la entidad de una manera diferente gracias a las identidades presentadas. [40]

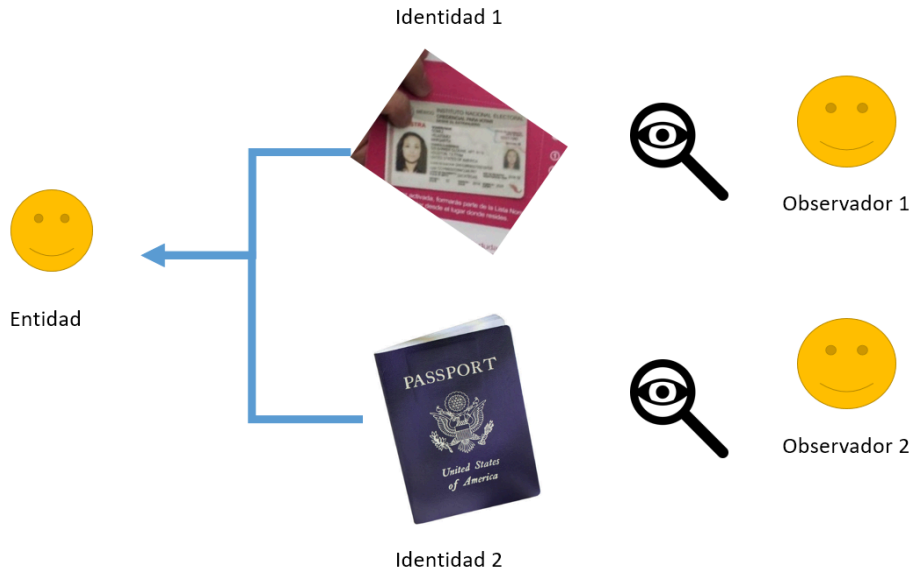

Imagen 27: Representación gráfica sobre la identidad de una entidad.

3.7.1. Single Sign On

Single sign on por sus siglas SSO es un mecanismo de autentificación. Este esquema permite a los usuarios iniciar sesión utilizando un conjunto único de credenciales. Esto permite navegar a los usuarios por diferentes aplicaciones, servicios con la misma credencial y sin necesidad de volverse a autentificar.
En pocas palabras, SSO permite al usuario autenticarse una sola vez y luego que la sesión es válida, también será válida para el resto de aplicaciones que hacen uso de SSO ver imagen 28. Por ejemplo, ingresando a Gmail podemos acceder a sus diferentes utilidades web, como Google Drive, Google Maps, Google Play Store, etc. SSO se usa comúnmente para la autenticación de empresas, portales de estudiantes, servicios de nube pública, banca y comercio electrónico.

SSO establece una relación de confianza entre una aplicación, servicio y un proveedor de identidades mejor conocido como (IdP). Para lograr esto, se siguen una serie de pasos de autenticación, validación y comunicación entre la aplicación y un servicio SSO centralizado.

SSO No es escalable, trabaja bien cuando es una sola compañía. Pero hay muchos procesos de negocio que requieren tener más de una compañía. Cuando un usuario necesita acceder a recursos en otra organización, surgen problemas.
Al pasar del tiempo con el surgimiento de las redes sociales y blogs, se multiplican usuarios, contraseñas, información de perfil (correo, foto, nombre, apellidos, direcciones).
Justo en este caso, se puede repetir la base de datos para acceder a diferentes sitios. La sincronización de dichas bases de datos, el espacio requerido para cada base de datos así como su mantenimiento son algunos de los argumentos en los cuales era necesario de alguna manera compartir esta información. Es aquí donde toman relevancia los estándares OAuth2 y OpenID.

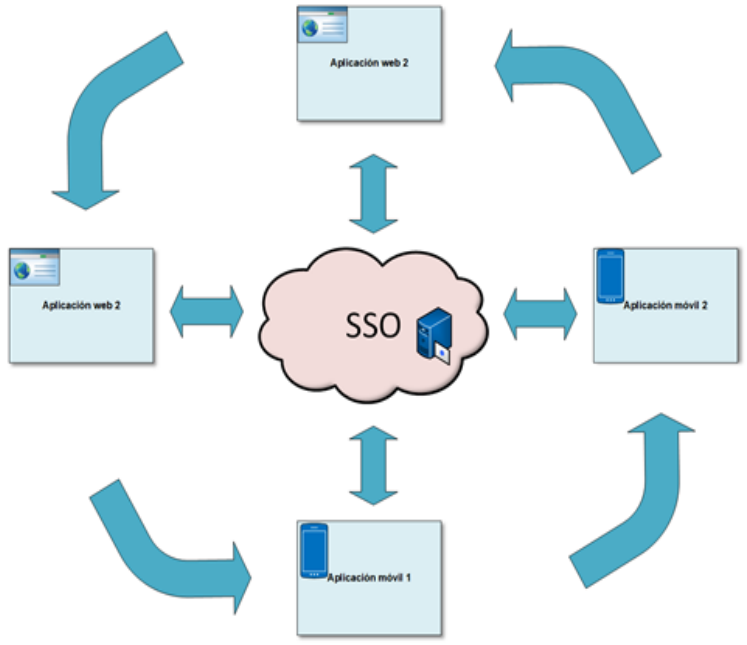

Imagen 28: Diagrama SSO, un único inicio de sesión permite acceder a diversas aplicaciones de un sistema.

La idea SSO de inicio de sesión único permanece pero actualmente ya es posible que se implemente con diferentes protocolos y servicios de autenticación como SAML[19], OAUTH, OpenID Connect, LDAP[20], ADFS[21] [48].

3.7.2. OAUTH 1 / RFC 5849

Oauth 1 fue un estándar abierto para la autorización delegada, actualmente se encuentra obsoleto. Este protocolo fue creado originalmente por una pequeña comunidad de desarrolladores web. Estos desarrolladores deseaban resolver el problema común de permitir acceso delegado a recursos protegidos. La versión 1.0 del protocolo OAuth sale en Octubre de 2007, para 2009 fue revisado. [41]
Este protocolo proporcionabá un mecanismo con el cual los clientes accedian a un recurso protegido del servidor, en nombre del propietario de dichos recursos. También provee un flujo para que usuarios finales autoricen a un tercero, acceso a los recursos de un servidor sin compartir sus credenciales (típicamente nombre de usuario, contraseña). OAuth 1 no es compatible con casos de uso de aplicaciones móviles, no brinda soporte nativo para la renovación de tokens y es más complejo.

[19] SAML (Security Assertion Markup Language) es un protocolo estándar para realizar un intercambio seguro de información para realizar la autenticación y autorización entre los diferentes dominios de confianza.
[20] LDAP (Lightweight Directory Access Protocol) es un protocolo de aplicación utilizado para acceder y mantener información en un servicio de directorio. LDAP es ampliamente utilizado para la autenticación y autorización en sistemas de directorio, como Microsoft Active Directory.
[21] ADFS (Active Directory Federation Services) es un servicio de federación de identidad proporcionado por Microsoft como parte de su plataforma de servicios de Active Directory.

3.7.3. OAUTH 2/RFC 6749

Un aspecto importante de OAUTH 2 y cabe resaltar es que “No es un protocolo de inicio de sesión”. Es un estándar que define **flujos de autorización** pero deja abierta la parte de **identificación** y **autenticación**. Este estándar proporciona una forma para que los desarrolladores ofrezcan sus servicios a través de una API sin obligar a sus usuarios a exponer sus contraseñas (y otras credenciales). Justo en este punto es donde se usan los tokens para acceder a los recursos. *Los tokens tienen un periodo de vida corta, son válidos por un cierto tiempo esto ayuda a la seguridad ya que no se comparten usuario y contraseña más que al inicio de sesión para autenticarse.*

Principales Roles en OAUTH 2.0 [42]
Durante el proceso de autorización se cuenta con los siguientes roles. ver imagen 29:

- El **cliente:** es la aplicación que está intentando conseguir el acceso a la cuenta de los usuarios.
- El **dueño del recurso:** es la persona que da el acceso a alguna parte de su cuenta.
- El **Servidor de autorización:** El servidor que emite el token de acceso para el cliente, después de una autenticación exitosa.
- El **Servidor de recursos:** El servidor que hospeda los recursos protegidos, capaz de aceptar y responder para proteger la solicitud de recursos protegidos usando tokens de acceso.

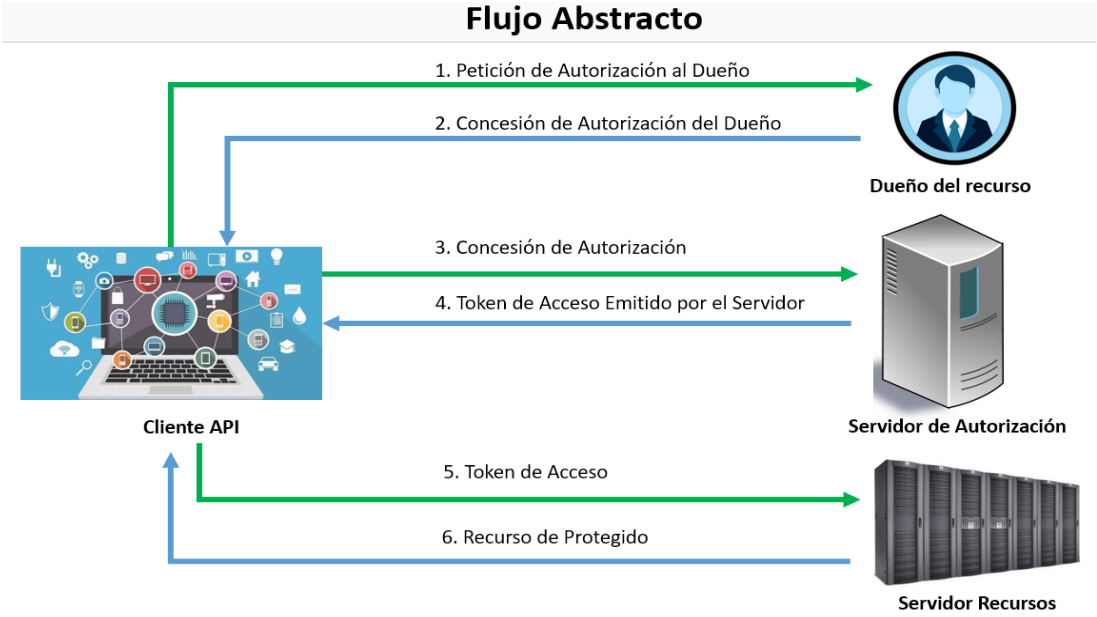

Imagen 29: Flujo abstracto de autorización de OAuth 2.0: Una representación visual del intercambio de información entre cliente, servidor de autorización y el servidor de recursos.

Una vez visto el diagrama general OAuth2 cabe mencionar que hay ciertos escenarios "The OAuth 2.0 Authorization Framework" RFC 6749 [49] Se definen cuatro tipos de concesión/escenarios:

1. Authorization Code Grant (Sección 4.1): Se utiliza para obtener tanto tokens de acceso como tokens de actualización y está optimizado para clientes confidenciales. Dado que se trata de un flujo basado en redireccionamiento, el cliente debe interactuar con el agente de usuario del propietario de los recursos (generalmente un navegador web) y recibir solicitudes entrantes (a través de redireccionamiento) desde el servidor de autorización.

2. Implicit Grant (Sección 4.2): Se utiliza para obtener tokens de acceso en clientes públicos que operan con una URI de redireccionamiento específica. Estos clientes suelen estar implementados en un navegador utilizando JavaScript. El cliente debe poder interactuar con el navegador web del propietario de los recursos y recibir solicitudes entrantes del servidor de autorización a través de redireccionamiento.

3. Resource Owner Password Credentials Grant (Sección 4.3): Es adecuado cuando *existe una relación de confianza entre el cliente y el propietario de recursos.* Permite que el cliente obtenga las credenciales (nombre de usuario y contraseña) del propietario de recursos, para obtener el token de acceso. También se utiliza para migrar clientes existentes que utilizan autenticación directa a OAuth. Sin embargo, su uso debe ser cuidadoso y solo permitido cuando otros flujos no son viables.

4. Client Credentials Grant (Sección 4.4): El cliente puede solicitar un token de acceso utilizando sus credenciales de cliente o medios de autenticación admitidos. Esto ocurre cuando el cliente solicita acceso a recursos protegidos bajo su control o a recursos de otro propietario de recursos previamente acordados con el servidor de autorización.

Además de estos flujos definidos en el RFC 6749, existen extensiones de OAuth 2.0, como el "OAuth 2.0 Device Authorization Grant" (RFC 8628) [50].
Este flujo se utiliza en escenarios donde el cliente no puede interactuar directamente con el agente de usuario del propietario de los recursos, como en dispositivos con pantallas pequeñas o sin capacidades de entrada de texto. Justo esta extensión de oauth 2.0 "Device flow" es el tipo de flujo de autorización que se utiliza para el proyecto ver imagen 30.

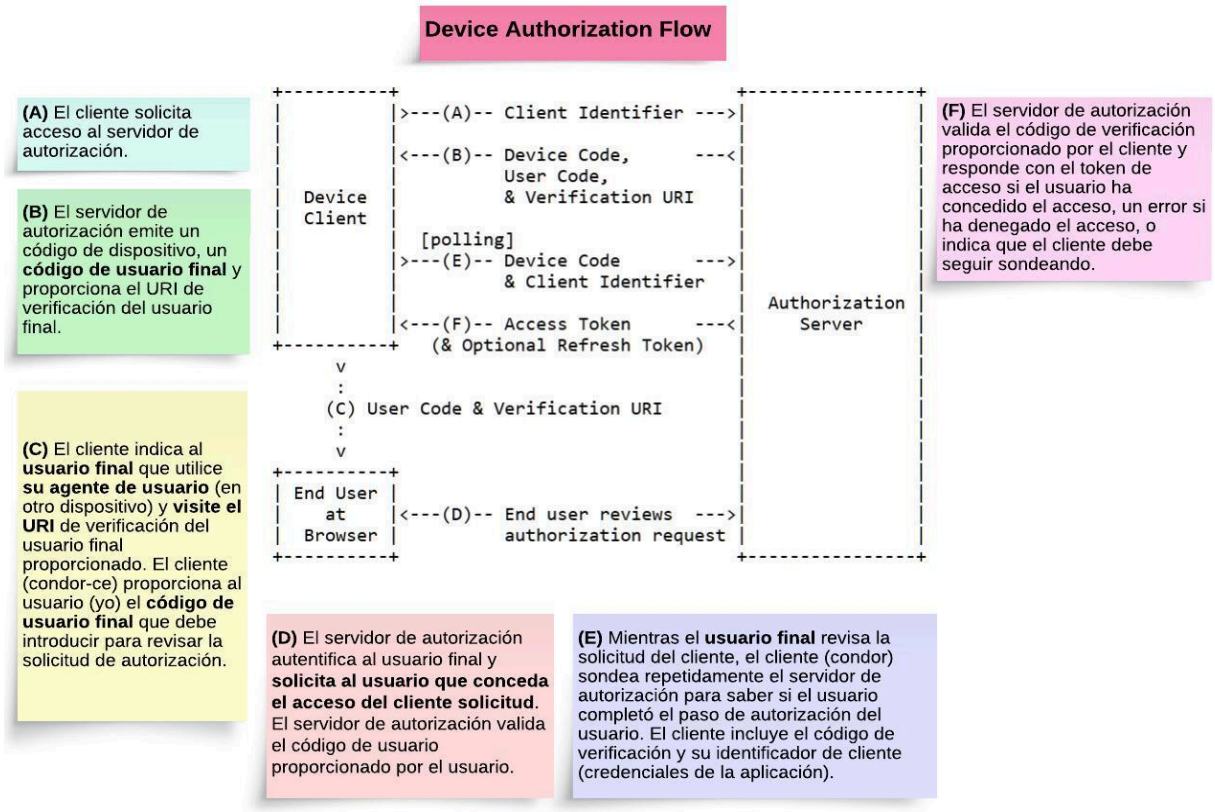

Imagen 30: Diagrama “Device Authorization Flow”.

Recordemos que OAuth2 no fue diseñado para implementar operaciones de inicio de sesión. Para los proveedores la mayoría solo exponen OAuth como una opción para delegar autorización con sus APIs. Los desarrolladores aprovecharon los flujos de autorización OAuth2 y así lograr un tipo de inicio de sesión.

En la práctica se puede implementar para crear y guardar una cookie de inicio de sesión [40] lo cual no es recomendable. El ejemplo práctico se muestra en el siguiente diagrama ver imagen 31.

LinkedIn puede acceder a los servicios de Gmail sin que ningún usuario sea autenticado por medio de un token de acceso (1). LinkedIn utiliza el token de acceso para acceder a las API de Gmail (2). Esto verifica que la persona que interactúa con la app “LinkedIn” efectivamente tiene cuenta válida de GMAIL. Para mantener la autenticación del usuario, LinkedIn puede implementar la creación y el almacenamiento de una cookie de sesión. Esta cookie se guarda (3) en el dispositivo del usuario y se utiliza para reconocerlo en futuras interacciones sin necesidad de repetir el proceso de autenticación.

La aplicación aceptará el token válido sin hacer preguntas, ya que previamente paso el proceso de autentificación. Esto podría ser peligroso, abren la puerta a este tipo de ataques: Confused Deputy attack[22].

[22] El ataque del "Confused Deputy" se produce cuando un programa o componente confía en otro para realizar acciones específicas, pero ese otro componente es manipulado por un atacante malintencionado. El atacante

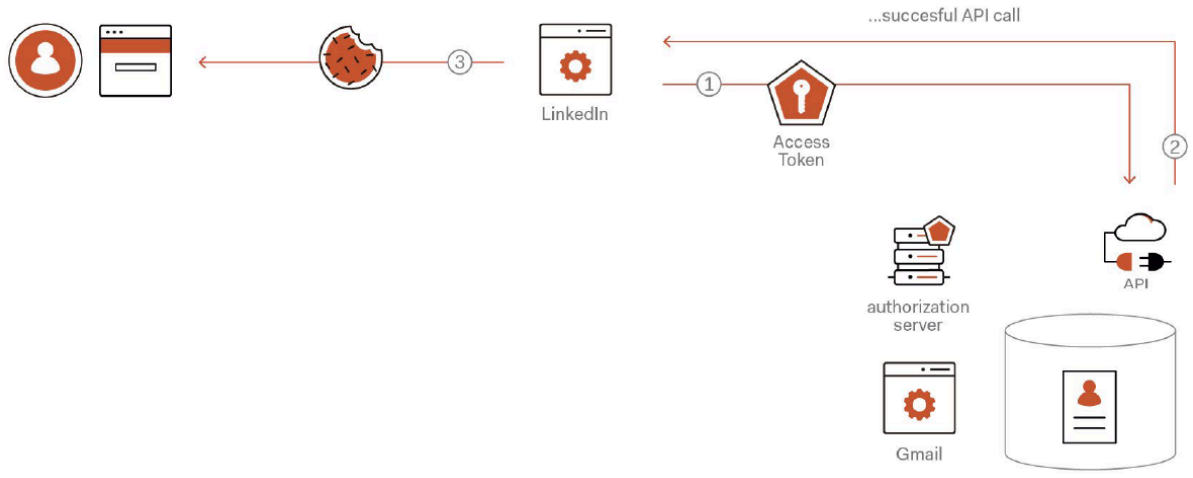

Imagen 31: Acceso a los recursos de Gmail utilizando un token de acceso.

3.7.4. OpenID Connect

A partir de utilizar Oauth 2 como inicio de sesión, se reunieron los principales de la industria y decidieron introducir una nueva especificación llamada **OpenID Connect**. Esta es una capa que se apoya con el protocolo OAuth2.0 [43]. Esto introduce otro nuevo artefacto llamado ID Token. El servidor de autorización puede emitir un token de identificación mediante los flujos ya definidos por OAuth2.

OpenID Connect es un protocolo de autenticación capaz de permitir a los usuarios iniciar sesión. Describe cómo las aplicaciones pueden solicitar un token de identificación. Este token ID es emitido por un servidor de autorización por los flujos definidos por OAuth2.
Las aplicaciones basadas en openID Connect dependen de los proveedores de identidad para gestionar los procesos de autenticación y para verificar las identidades (es decir, los atributos personales) de sus usuarios.

Al utilizar OpenID Connect, en lugar de ocuparse de las credenciales de estos usuarios, su aplicación puede delegar el proceso de autenticación a un **proveedor de identidad** (por ejemplo, Google, Microsoft o Auth). Cuando los visitantes inicien este proceso, su aplicación los redirigirá al proveedor de identidad de su elección, donde se autentican para demostrar su identidad.

engaña al componente confiado para que realice acciones no autorizadas o no deseadas en lugar de seguir las instrucciones legítimas.

Esto implica que se debe de autenticar el usuario antes de acceder a los recursos. De esta manera no se confía ciegamente en el token de acceso, el cual puede ser utilizado por un tercero para acceder a los servicios o recursos.

Para ejemplificar la idea (ver imagen 32) tenemos un diagrama de flujo, linkedin presenta las credenciales del cliente (1) frente al servidor de autorización el cual al validar dichas credenciales (2) del usuario y de ser correctas proporciona el token ID y Token de acceso. Por último se almacenan en las cookies (3) para poder acceder a los recursos posteriormente [40].

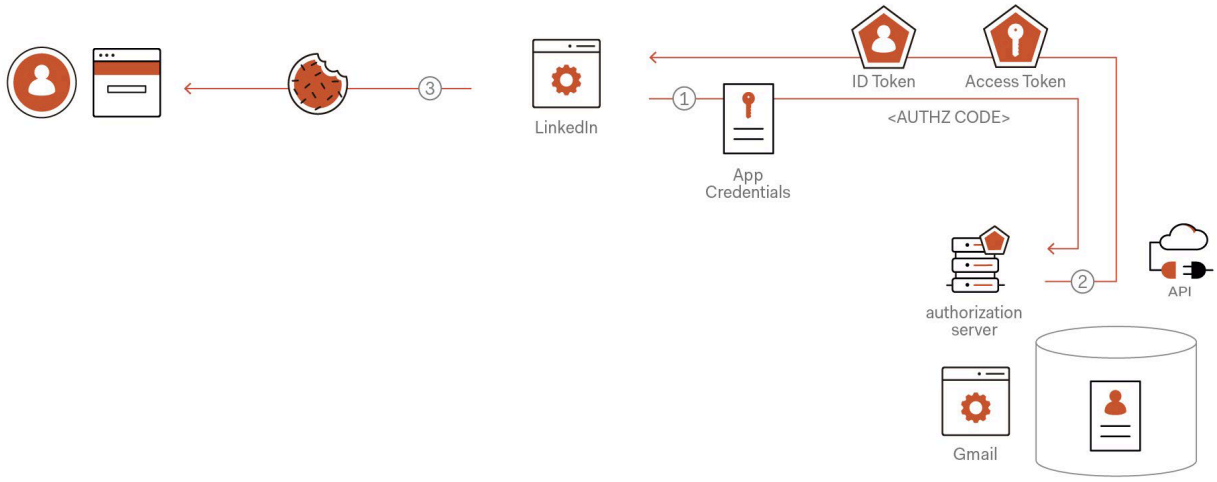

Imagen 32: Acceso a los recursos de Gmail utilizando ID token (Autenticación) y Token de acceso (Autorización).

En resumen OIDC [23]define cómo los proveedores de identidad y las aplicaciones interactúan para establecer la autenticación del usuario final de forma segura.

¿Por qué hay que preocuparse por un protocolo de autenticación? Principalmente porque los desarrolladores quieren resolver la gestión de la identidad de forma segura e interoperable.

[23] OpenID Connect, popularmente abreviado como OIDC.

Casos de uso de OpenID Connect

Los protocolos en los que se puede aprovechar este escenario son tres:

1. Puedes utilizar OpenID Connect para que tus usuarios puedan reutilizar sus cuentas en un proveedor de identidad. Es decir, en lugar de pedir a los usuarios que creen otra cuenta, lo que significa pedirles que recuerden otro conjunto de credenciales, podrías aprovechar de OIDC para integrarse con un proveedor de identidades como Google o Microsoft y permitirles reutilizar las cuentas existentes.

2. Cuando el protocolo se utiliza para crear un centro de proveedores de identidad. En este escenario, en lugar de hacer que tu aplicación se comunique con múltiples proveedores, puedes hacer que se conecte a uno solo que funcione como un centro para los demás.

3. Cuando funciona como un proxy para otros protocolos. Por ejemplo, puede hacer que un proveedor de identidad de OpenID Connect funcione como proxy para un protocolo más restrictivo como SAML[24]. Utilizando este enfoque, puedes hacer que un dispositivo con recursos limitados se integre con un proveedor de identidad SAML a través de “OpenID Connect".

OpenID Connect proporciona un abanico de posibilidades capaces de ayudarte a hacer la gestión de identidades más fácil y más extensible.

3.7.5. OAUTH & OpenID Connect

Como hemos visto, OAuth 2.0 y OpenID Connect están estrechamente relacionados y son utilizados ampliamente en aplicaciones web como servicios. Mientras OAuth 2.0 se centra principalmente en la autorización de acceso a recursos protegidos. OpenID Connect se basa en OAuth 2.0 y agrega una capa de autenticación a la autorización proporcionada por OAuth 2.0.

Esto permite a las aplicaciones obtener tanto tokens de acceso para acceder a recursos protegidos como información de identidad verificada sobre el usuario autenticado, todo en un solo flujo ver imagen 33.

[24] SAML (Security Assertion Markup Language) es un estándar de intercambio de información de autenticación y autorización en entornos de seguridad informática. Proporciona un marco para compartir declaraciones de identidad entre entidades, como proveedores de identidad y proveedores de servicios.

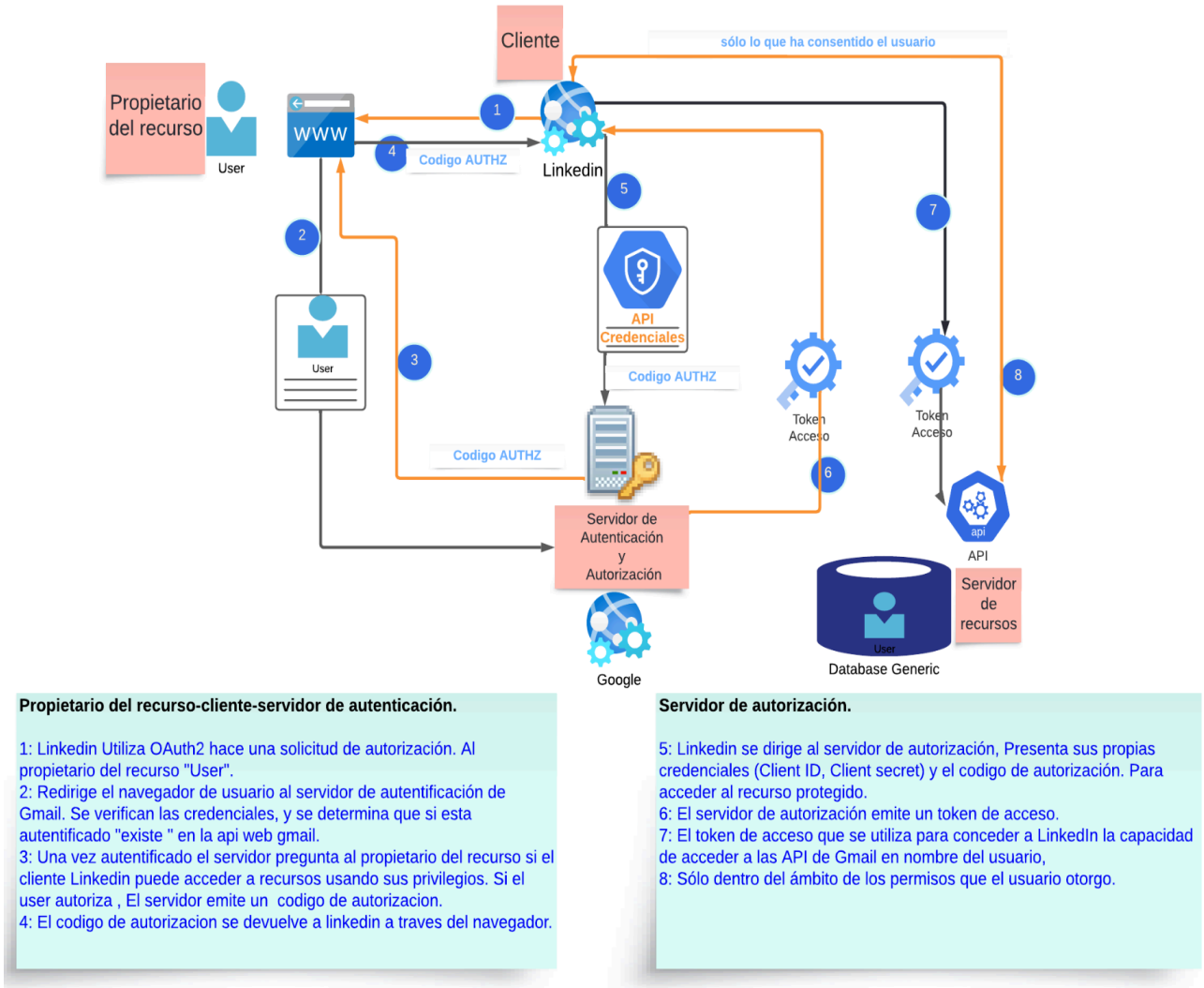

Imagen 33: Diagrama de Autenticación / Autorización Oauth 2 y OIDC.

El número de especificaciones que se interrelacionan para hacer una implementación es muy amplio, pero muy importante si se quiere hacer una correcta implementación, desarrollo de alguna aplicación. En seguida se muestran algunas referentes a OAUTH2, JWT y OpenID. ver imagen 34.

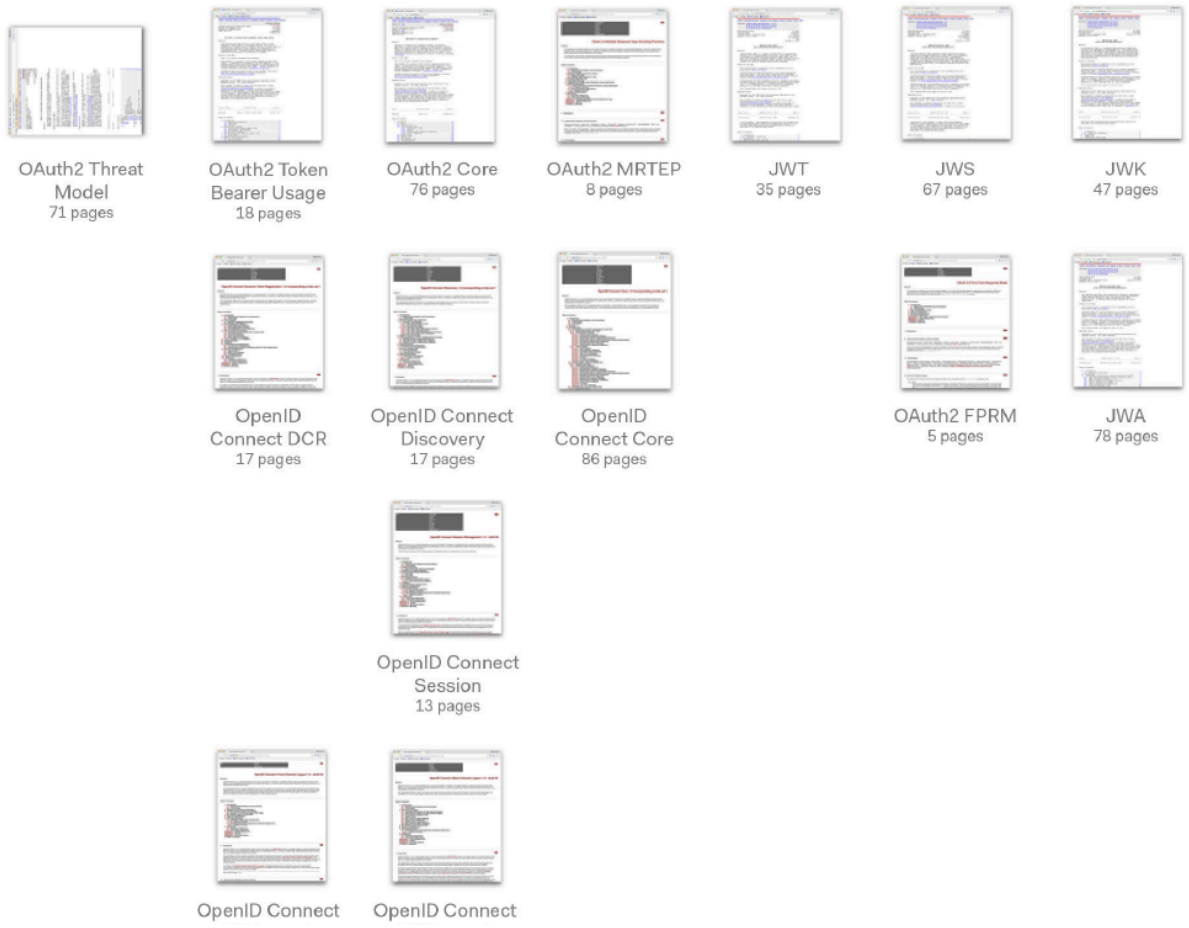

Imagen 34: Especificaciones entorno a la implementación OAUTH 2, JWT y OIDC.

3.8. IAM INDIGO

INDIGO IAM (Identity and Access Management) es un servicio de Gestión de Identidad y Acceso desarrollado por primera vez en el contexto del proyecto INDIGO-Datacloud Horizonte 2020, financiado por la Comisión Europea. Actualmente mantenido y desarrollado por el Istituto Nazionale di Fisica Nucleare (INFN) [44]. El proyecto se centra en el desarrollo de una plataforma de software completamente gratuita, de código abierto que pueda operar en infraestructuras de red públicas y privadas. Esta plataforma permitirá a los investigadores acceder de manera sencilla a recursos compartidos de computación y almacenamiento. Por esta razón ha sido seleccionado por WLCG para ser el núcleo del servicio de autorización de próxima generación en apoyo a la informática del Gran Colisionador de Hadrones (LHC).

Este servicio IAM es desplegado para una gran comunidad de usuarios que comparten recursos, siguiendo el concepto de Organización Virtual (VO). Se integra con aplicaciones y servicios de clientes a través de mecanismos estándar de **OAuth/OpenID Connect.** Es básicamente un proyecto que implementa los conceptos previamente vistos y ahí radica su importancia.

El Servicio de Gestión de Identidad y Acceso (IAM) de INDIGO proporciona una solución para el cómputo científico. INDIGO IAM proporciona una capa donde las identidades, inscripciones, pertenencia a grupos y otros atributos y políticas de autorización sobre recursos distribuidos pueden ser gestionados por INDIGO [45].

Dentro de las funciones principales tenemos:

- **Autentificación**: IAM proporciona mecanismos de autentificación como SAML, X.509, OpenID Connect.
- **Manejo de sesiones**: IAM proporciona funcionalidad de gestión de sesiones. Las sesiones se utilizan para proporcionar la funcionalidad de inicio y cierre de sesión únicos a las aplicaciones y servicios cliente.
- **Inscripción**: IAM proporciona funciones de inscripción y registro para que los usuarios puedan unirse a grupos.
- **Gestión de atributos e identidades**: IAM proporciona servicios para gestionar pertenencia a grupos, asignación de atributos a los administradores de grupos.
- **Definición, distribución y evaluación de políticas**: la IAM proporciona herramientas y API para definir políticas de autorización sobre recursos distribuidos.

IAM es una aplicación Java **Spring Boot**[25]. Normalmente desplegada detrás de **NGINX**[26]. IAM almacena datos en una base de datos MariaDB/MYSQL. IAM puede emitir tokens compatibles con el perfil SciTokens. Dentro de los requisitos previos para usar INDIGO necesitará:

- Un certificado X.509, utilizado para la terminación de SSL en el proxy inverso NGINX; puede obtener uno gratis de Let's Encrypt;
- Un servidor NGINX configurado para actuar como un proxy inverso[27] para la aplicación web de IAM;
- Una instancia de base de datos MariaDB/MySQL preferentemente desde un contenedor usando dockers o podman;
- Un almacén de claves JSON (keystore) que contiene las claves utilizadas para firmar tokens web JSON;

[25] Spring Boot es un framework de desarrollo de aplicaciones Java que simplifica y agiliza el proceso de creación de aplicaciones empresariales.

[26] NGINX Es un servidor Web/proxy inverso con prestaciones de alto rendimiento y código abierto. Se utiliza ampliamente como servidor HTTP para entregar contenido web estático y dinámico, así como para actuar como proxy inverso para equilibrar la carga del tráfico entrante hacia múltiples servidores.

[27] Un proxy inverso se ubica en el lado del servidor y desempeña el papel de intermediario entre los clientes y un servidor web. Su función principal es recibir las solicitudes de los clientes en nombre de uno o más servidores, y posteriormente transmitir esas solicitudes a los servidores correspondientes.

3.8.1. Arquitectura IAM

La arquitectura INDIGO IAM está basada en componentes básicos "Arquitectura de microservicios" ver imagen 35. Este tipo de arquitectura de microservicios, así como el despliegue por medio de contenedores[28] como dockers[29] ha permitido la implementación de indigo con otros sistemas y algunas integraciones de INDIGO IAM con componentes comerciales ver imagen 36. [44]

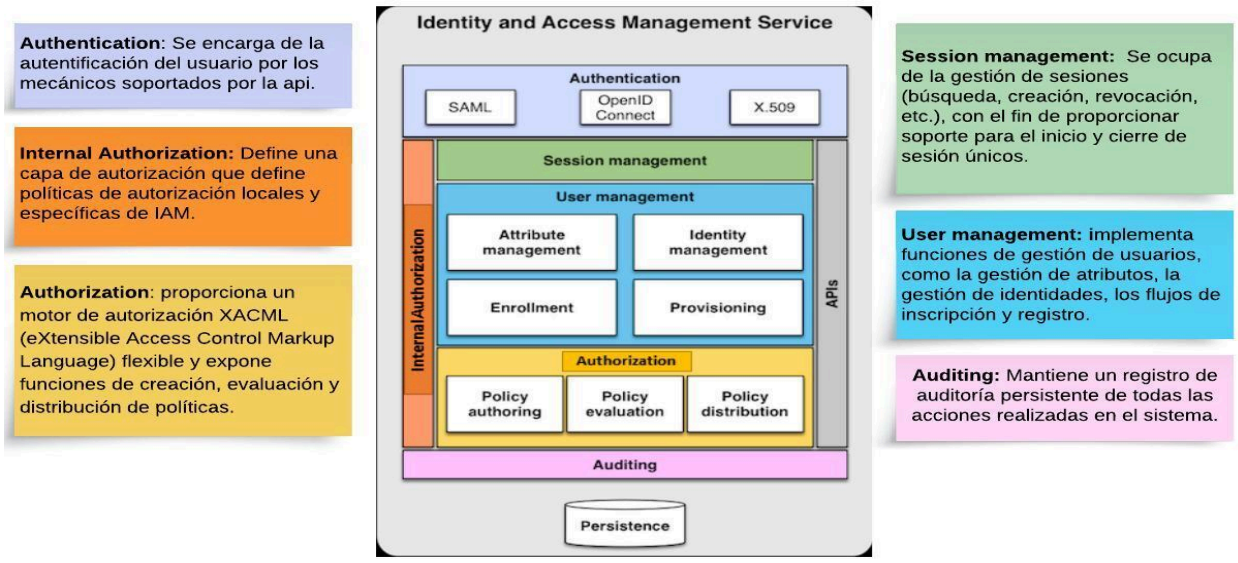

Imagen 35: Arquitectura INDIGO IAM basada en microservicios.

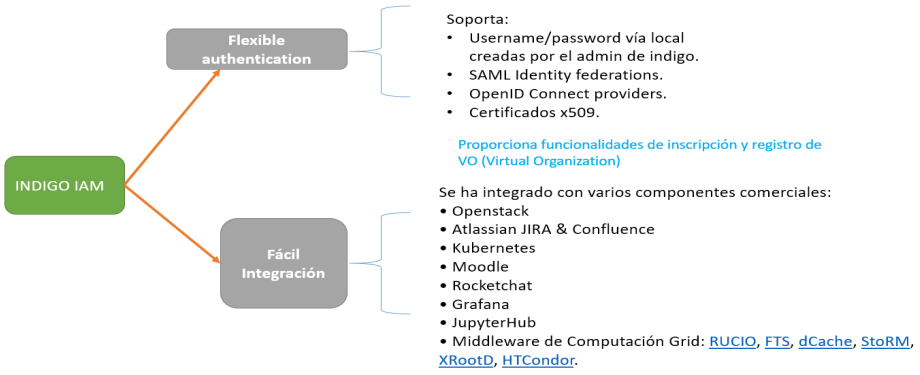

Imagen 36: Autentificación e integración de INDIGO IAM

[28] Los contenedores: son unidades de software empaquetadas que incluyen el código de la aplicación y todas sus bibliotecas y dependencias necesarias, permitiendo que la aplicación se ejecute de manera consistente en cualquier entorno.

[29] Docker es una herramienta de código abierto. Simplifica la distribución y ejecución de aplicaciones encapsuladas en contenedores de software.

3.9. Clúster

"El concepto de Cluster se puede definir como un sistema de procesamiento (paralelo o distribuido) conformado por un conjunto de computadoras independientes (nodos), interconectadas entre sí y que funciona como un solo recurso computacional. Otro componente básico en un cluster es la interfaz de la red, la cual es responsable de transmitir y recibir los paquetes de datos, que viajan a través de la red entre los nodos"[46]. Ver imagen 37

Los clusters son empleados en una gran variedad de aplicaciones y ramas: La investigación científica, el análisis de grandes conjuntos de datos, la renderización de gráficos y la simulación de procesos complejos.
Los clusters pueden ser diseñados para ser altamente redundantes y tolerantes a fallas, lo que los hace ideales para aplicaciones críticas que requieren alta disponibilidad y tiempo de actividad constante.

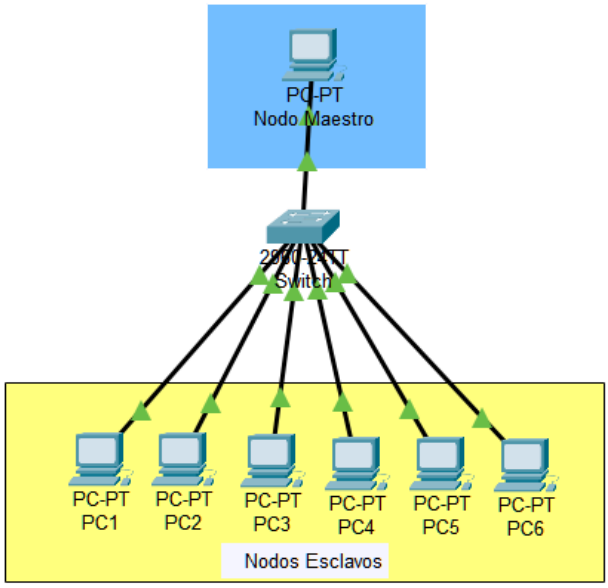

Imagen 37: Esta imagen representa las partes fundamentales de un clúster. El nodo maestro, encargado de la gestión y coordinación de las tareas, se comunica con los nodos esclavos a través de un switch y una red de alta velocidad. Esta arquitectura permite distribuir la carga de trabajo y mejorar el rendimiento en aplicaciones de alto rendimiento y tareas computacionales intensivas.

4. Implementación de la Maqueta

En este capítulo se mencionará el proceso de implementación de la maqueta (ver tabla 13) basado en el trabajo de investigación y los conceptos previamente vistos. Esta sección se divide en tres componentes principales, cada una de las cuales representa un componente[30] esencial de la maqueta: el Cluster con Slurm, HTCondor-CE e INDIGO IAM (ver imagen 38).

1. Cluster con Slurm: Se creó un cluster con máquinas virtuales. El cual nos brindará el poder de cómputo para procesar trabajos y es administrado por el sistema batch slurm.
2. HTCondor-CE: Es un software instalado en un Host que funge la función de enviar y administrar trabajos en su sistema de lotes local "slurm".
3. INDIGO IAM: Es un servicio de Gestión de Identidad y Acceso. En el cual recae la autorización para el uso de los recursos computacionales por medio de JWT.

Maqueta		
Nombre de los Nodos	**S.O**	**Hardware**
nodo 1	**CentOS 7**	**Model name:** **Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz** **CPU(s):** **4** **Ram:** **7.6G** **Almacenamiento:** **3.9 G**
nodo 2	**CentOS 7**	**Model name:** **Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz** **CPU(s):** **4** **Ram:** **7.6G** **Almacenamiento:** **3.9 G**
nodo 3	**CentOS 7**	**Model name:** **Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz** **CPU(s):** **4** **Ram:** **7.6G** **Almacenamiento:** **3.9 G**
maestro	**CentOS 7**	**Model name:** **Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz** **CPU(s):** **4** **Ram:** **7.6G** **Almacenamiento:** **3.9 G**
submit	**CentOS 7**	**Model name:** **Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz** **CPU(s):** **4** **Ram:** **7.8G** **Almacenamiento:** **3.9 G**
iam	**Debian 11**	**Model name:** **Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz** **CPU(s):** **2** **Ram:** **5.8G** **Almacenamiento:** **2.9 G**

Tabla 13: Nodos que componen la maqueta y algunas de sus características.

[30] Componente: representa una sección funcional de la maqueta, esta puede contener uno o más nodos.

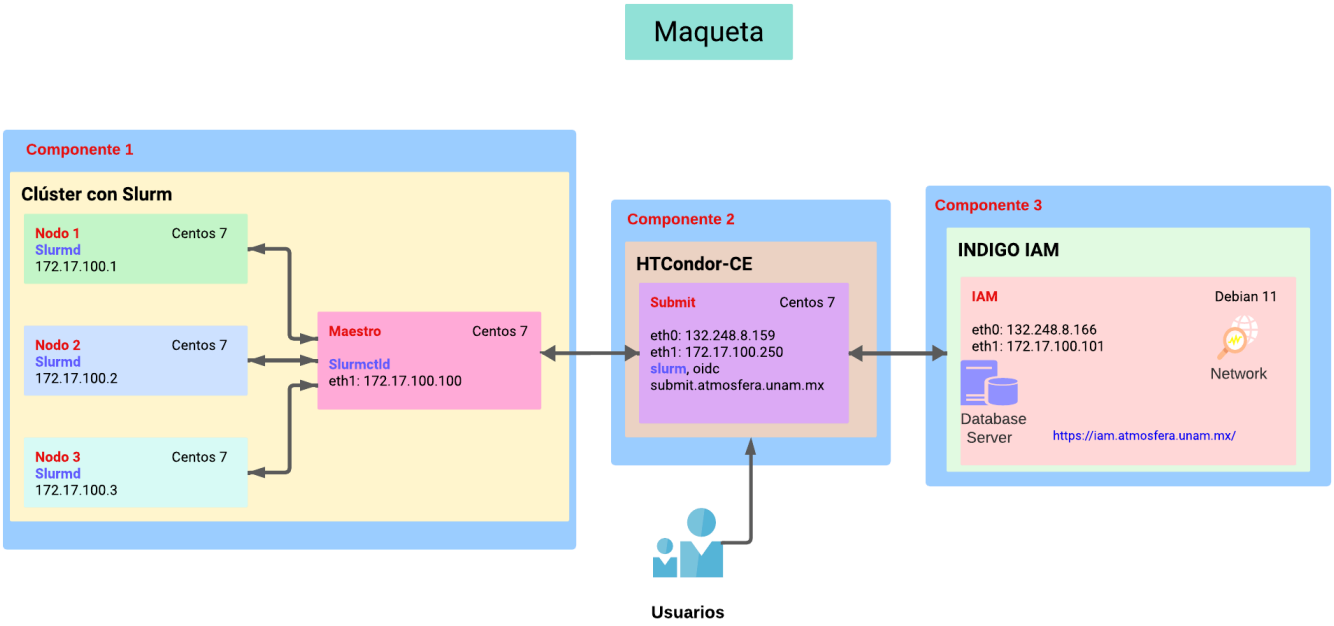

Imagen 38: Diagrama general de la Maqueta Grid.

4.1. Clúster con Slurm

Uno de los elementos principales de la maqueta es tener un cluster funcional con el sistema Slurm.
A Continuación se definen los conceptos de cluster y slurm.

- El concepto de cluster se define como un sistema de procesamiento conformado por un conjunto de computadoras interdependientes (nodos), que están interconectadas entre sí y funcionan como un solo recurso computacional.

- Slurm es una herramienta para distribuir tareas computacionales entre nodos disponibles en sistemas HPC, HTC.

Tener un clúster funcional implica que las computadoras tengan una configuración adecuada, que las conexiones entre los nodos estén establecidas para garantizar su comunicación e instalación del software necesario para lograr el funcionamiento del clúster (Slurm).

Nuestro cluster cuenta con tres nodos esclavos y un nodo maestro (ver imagen 38). Como las primeras configuraciones encontramos la actualización del software, paquetes, establecer correctamente la hora, probar que todos los nodos de la maqueta tengan acceso entre sí. En la tabla 15 listamos las configuraciones que fueron necesarias para obtener esta primera parte de la maqueta funcional.

4.1.1. Configuración inicial del clúster

Una de las configuraciones iniciales es actualizar cada uno de los nodos (yum update, yum upgrade), verificar si tienen la fecha y hora correcta. El comando timedatectl permite consultar y cambiar la configuración del reloj y zona horaria.

TodosLosNodos# timedatectl

TodosLosNodos# timedatectl set-timezone America/Mexico_City

Modificamos el archivo /etc/hosts. El cual es un archivo de configuración en sistemas Unix y Linux. Su principal función es asociar nombres de host a direcciones IP en la resolución de nombres en la red local sin depender de un servidor DNS externo. Este archivo hosts está en todos los nodos de la maqueta. Archivo **/etc/hosts** contiene las siguientes líneas**:**

```
127.0.0.1   localhost
132.248.8.166   iam.atmosfera.unam.mx   iam
172.17.100.100  maestro
172.17.100.1    nodo1 n1
172.17.100.2    nodo2 n2
172.17.100.3    nodo3 n3
172.17.100.250  submit htc
```

Ahora generamos las claves pública y privada necesarias para autenticarse en un sistema remoto a través de SSH. El comando **ssh-keygen -t rsa** se utiliza en Linux para generar pares de claves RSA[31] para su uso en el protocolo SSH (Secure Shell). SSH es un protocolo de red que permite una comunicación segura y encriptada entre dos sistemas, y es ampliamente utilizado para acceder de forma remota a servidores y dispositivos. Al evitar el uso de contraseñas se evitan el uso de contraseñas débiles que sean vulnerables a ataques de fuerza bruta. También facilita la administración de usuarios y el acceso a múltiples nodos sin la complejidad de administrar y recordar contraseñas para cada uno de los nodos. Ver imagen 39.

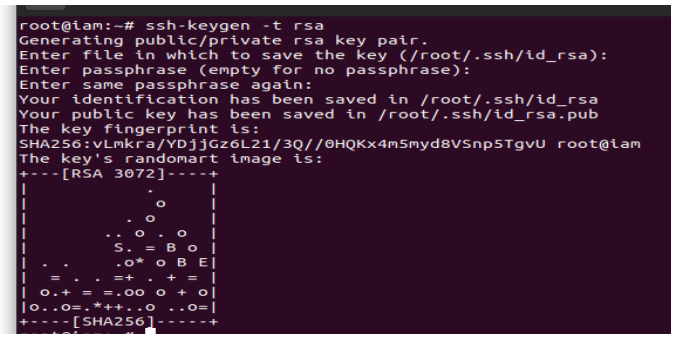

Imagen 39: Generación de claves RSA en el nodo iam.

[31] RSA (Rivest, Shamir y Adleman) Es un algoritmo de cifrado asimétrico ampliamente utilizado en criptografía. Se basa en el uso de una pareja de claves, una pública y una privada, que se utilizan para cifrar y descifrar datos.

Después la generación de la llave se difunde al resto de nodos. ssh-copy-id Se utiliza para copiar una clave pública SSH al archivo ***~/.ssh/authorized_keys*** en un servidor remoto. Para poder acceder por ssh a los nodos desde iam.

Oscar@iam **#** ssh-copy-id root@172.17.100.100 //maestro
Oscar@iam **#** ssh-copy-id root@ 172.17.100.1 //nodo1
Oscar@iam **#** ssh-copy-id root@ 172.17.100.2 //nodo2
Oscar@iam **#** ssh-copy-id root@ 172.17.100.3 //nodo3

Este mismo proceso de generar la llave "**ssh-keygen -t rsa**" y copiarla al resto de nodos se hace con cada nodo. En pocas palabras el nodo maestro genera su llave y se copia al resto de los nodos (nodos esclavos) de igual manera con el nodo 1, nodo 2 y nodo 3.

Ahora creamos una cuenta de usuario sin privilegios llamado "usuarioNormal" que servirá para probar el funcionamiento con slurm.

NodoMaestro/esclavos # sudo useradd -u 1002 -m -c "Usuario normal" -d /home/usuarioNormal -s /bin/bash usuarioNormal

Después de esto vamos a compartir el directorio home del nodo maestro a los nodos de trabajo y el nodo submit por NFS[32].

NodoMaestro/esclavos/submit# yum -y install nfs-utils // Instala nfs

Una vez instalado el software en el nodo maestro editamos el archivo "/etc/exports" el cual es utilizado en sistemas Unix y Linux que ejecutan el servicio NFS (Network File System) para definir qué sistemas o hosts remotos tienen permiso para montar y acceder a sistemas de archivos compartidos a través de NFS desde el servidor NFS. El archivo exports queda de la siguiente manera.

```
/home    172.17.100.250(rw,no_subtree_check,sync,no_root_squash,async)
/home    172.17.100.100(rw,no_subtree_check,sync,no_root_squash,async)
/home      172.17.100.1(rw,no_subtree_check,sync,no_root_squash,async)
/home      172.17.100.2(rw,no_subtree_check,sync,no_root_squash,async)
/ home 172.17.100.3(rw,no_subtree_check,sync,no_root_squash,async)
```

Después de cualquier cambio sobre el archivo /etc/exports tenemos que ejecutar el siguiente comando para actualizar la tabla nfs:

NodoMaestro# exportfs -v // muestra una lista detallada de los sistemas de archivos exportados.

Ahora Activamos, Iniciamos y vemos el estatus del servicio nfs en los nodos.

NodoMaestro/esclavos/submit# systemctl enable nfs // Activar nfs
NodoMaestro/esclavos/submit# systemctl start nfs // Iniciar nfs
NodoMaestro/esclavos/submit# systemctl status nfs // Ver status nfs

[32] NFS "Network File System" es un protocolo que permite compartir sistemas de archivos y acceder a ellos de forma remota a través de una red, facilitando el intercambio de datos y recursos entre computadoras en una red.

Para hacer el montaje del directorio /home editamos el archivo /etc/fstab el cual es un archivo de configuración utilizado en sistemas Unix y Linux para definir cómo y dónde se deben montar los sistemas de archivos durante el proceso de arranque del sistema.

NodosEsclavos/submit# nano /etc/fstab // al archivo fstab agregamos la linea:

```
maestro:/home /home nfs defaults 0 0
```

Ya realizado el cambio en el archivo fstab de los nodos esclavos y submit montamos el directorio. ver imagen 40.

NodosEsclavos/submit# mount -t nfs maestro:/home /home // montamos el directorio home
NodosEsclavos/submit# df -h // muestra las particiones

```
[root@nodo3 ~]# ssh n1
Last login: Sun Sep 10 20:43:40 2023 from 172.17.100.3
[root@nodo1 ~]# df -h
S.ficheros     Tamaño Usados  Disp Uso% Montado en
devtmpfs         3.9G      0  3.9G   0% /dev
tmpfs            3.9G      0  3.9G   0% /dev/shm
tmpfs            3.9G   8.6M  3.9G   1% /run
tmpfs            3.9G      0  3.9G   0% /sys/fs/cgroup
/dev/sda3        7.1G   1.8G  5.4G  25% /
/dev/sda1       1014M   221M  794M  22% /boot
maestro:/home     17G   1.7G   16G  10% /home
tmpfs            783M      0  783M   0% /run/user/0
[root@nodo1 ~]# ssh n2
Last login: Sun Sep 10 20:44:01 2023 from 172.17.100.1
[root@nodo2 ~]# df -h
S.ficheros     Tamaño Usados  Disp Uso% Montado en
devtmpfs         3.9G      0  3.9G   0% /dev
tmpfs            3.9G      0  3.9G   0% /dev/shm
tmpfs            3.9G   8.6M  3.9G   1% /run
tmpfs            3.9G      0  3.9G   0% /sys/fs/cgroup
/dev/sda3        7.1G   1.7G  5.4G  24% /
/dev/sda1       1014M   221M  794M  22% /boot
tmpfs            783M      0  783M   0% /run/user/0
maestro:/home     17G   1.7G   16G  10% /home
[root@nodo2 ~]# ssh n3
Last login: Sun Sep 10 22:10:54 2023 from 172.17.100.100
[root@nodo3 ~]# df -h
S.ficheros     Tamaño Usados  Disp Uso% Montado en
devtmpfs         3.9G      0  3.9G   0% /dev
tmpfs            3.9G      0  3.9G   0% /dev/shm
tmpfs            3.9G   8.7M  3.9G   1% /run
tmpfs            3.9G      0  3.9G   0% /sys/fs/cgroup
/dev/sda3        7.1G   1.7G  5.4G  24% /
/dev/sda1       1014M   221M  794M  22% /boot
maestro:/home     17G   1.7G   16G  10% /home
tmpfs            783M      0  783M   0% /run/user/0
```

Imagen 40: Montaje del directorio /home en los nodos esclavos y nodo submit.

4.1.2. Instalación y Configuración de Slurm

Antes de iniciar la instalación de slurm se debe instalar “epel-release” el cual es un paquete de repositorio que se utiliza en sistemas basados en Red Hat (como CentOS y RHEL) para habilitar el repositorio EPEL (Extra Packages for Enterprise Linux). Al instalar este paquete nos permitirá acceder a los repositorios epel para instalar slurm. También se debe instalar munge ya que es un prerrequisito al momento de instalar slurm.

Munge es un software utilizado para la autenticación segura y el cifrado de mensajes en sistemas distribuidos, como clústeres de computadoras de alto rendimiento. Munge proporciona una forma segura de autenticar y cifrar datos antes de transmitirlos a través de la red, lo que es esencial en entornos donde se necesita una alta seguridad en la comunicación entre nodos del clúster.

Instalación de munge en el nodo maestro, nodos esclavos y submit:

```
Maestro/esclavos/submit# yum update                    // Actualización
Maestro/esclavos/submit# yum install epel-release -y   // Repositorio epel
Maestro/esclavos/submit# yum -y install munge                  // Instalación
```

Se crea la llave de munge, dicha llave al generarse se ubica en /etc/munge/munge.key

```
Maestro# create-munge-key                    // creación de la llave munge
```

Una vez generada la llave en el nodo maestro, se copiará a los nodos de cómputo y submit respetando los permisos, usuario y grupo. Se verifican permisos en la imagen 41.

```
Maestro# scp /etc/munge/munge.key nodo1:/etc/munge/munge.key
Maestro# scp /etc/munge/munge.key nodo2:/etc/munge/munge.key
Maestro# scp /etc/munge/munge.key nodo3:/etc/munge/munge.key
Maestro# scp /etc/munge/munge.key submit:/etc/munge/munge.key
```

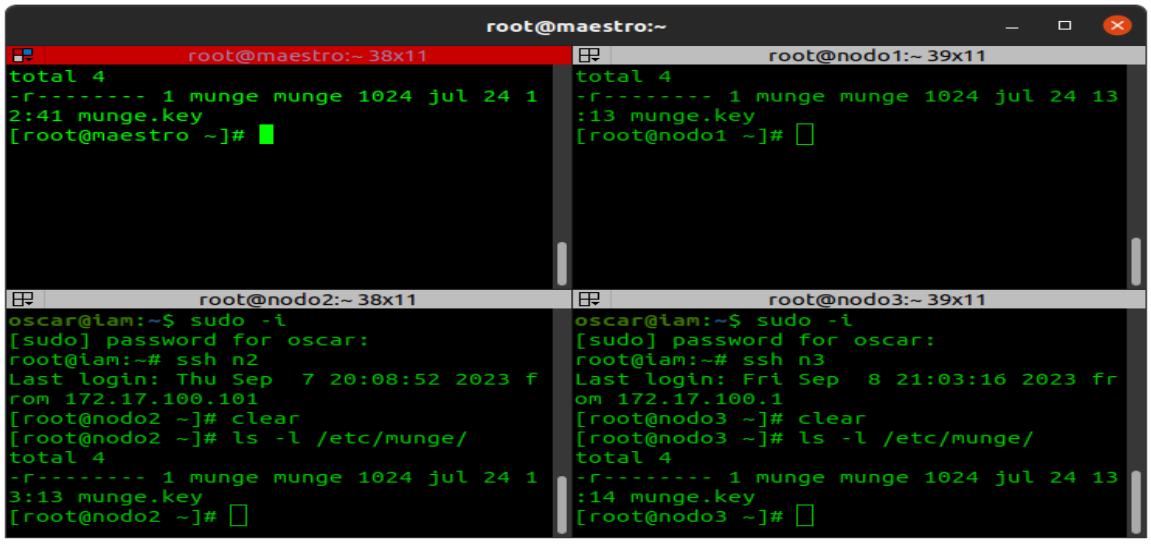

Imagen 41: Permisos de munge.key en los nodos.

Los siguientes comandos sirven para Habilitar, Iniciar y ver el estatus del servicio de munge en el nodo maestro, nodos esclavos y submit:

```
Maestro/esclavos/submit# systemctl enable munge.service     // Habilita el servicio munge
Maestro/esclavos/submit# systemctl start munge.service      // Inicia el servicio munge
Maestro/esclavos/submit# systemctl status munge.service     // ver el estado del servicio munge
```

4.1.2.1. Instalación de Slurm en el nodo Maestro

De acuerdo a la documentación de slurm, el nodo maestro usará el demonio **slurmctld** ver imagen 42. Que es el demonio de gestión central de slurm.

```
Maestro# yum -y install slurm-slurmctld     // Instalar slurmctld
Maestro# systemctl enable slurmctld         // Habilitar slurmctld
Maestro# systemctl start slurmctld          // Iniciar slurmctld
Maestro# systemctl status slurmctld         // Status de slurmctld
```

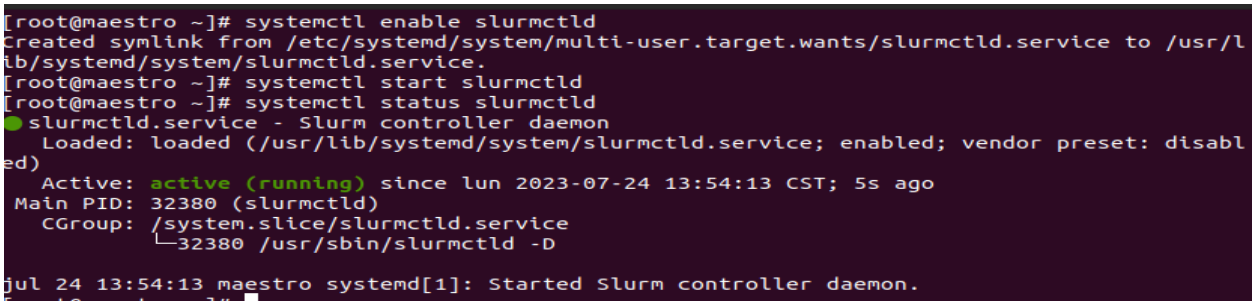

Imagen 42: Estatus del demonio Slurmctld.

Al instalar slurm se crea el archivo de configuración: **"etc/slurm/slurm.conf".**

slurm.conf es un archivo ASCII que describe información general de configuración de Slurm, los nodos que se administrarán, información sobre cómo se agrupan esos nodos en particiones y varios parámetros de programación asociados con esas particiones. Este archivo debe ser coherente en todos los nodos del clúster, debe pertenecer al grupo y ser propietario de slurm. [47]

El archivo *slurm.conf* debe ser legible por todos los usuarios de Slurm, ya que lo utilizan muchos comandos de Slurm.

El contenido del archivo no distingue entre mayúsculas y minúsculas, excepto los nombres de nodos y particiones. Cualquier texto que siga a un "#" en el archivo de configuración se trata como un comentario hasta el final de esa línea. El archivo de configuración se encuentra en el anexo.

Ahora copiamos el archivo de configuración de slurm del maestro a los nodos esclavos y submit:

```
Maestro# scp -r /etc/slurm/slurm.conf nodo1:/etc/slurm/slurm.conf
Maestro# scp -r /etc/slurm/slurm.conf nodo2:/etc/slurm/slurm.conf
Maestro# scp -r /etc/slurm/slurm.conf nodo3:/etc/slurm/slurm.conf
Maestro# scp -r /etc/slurm/slurm.conf submit:/etc/slurm/slurm.conf
```

4.1.2.2. Instalación de Slurm en los nodos [1-3] y Submit

Ahora, en los nodos esclavos del cluster, se procede a la instalación de Slurm. Esto implica la instalación del demonio Slurmd, conocido como "Slurm Daemon", el cual desempeña un papel crucial como uno de los componentes clave en el sistema de gestión de trabajos Slurm. Este demonio se ejecuta en cada nodo del clúster y asume la responsabilidad de administrar y coordinar los recursos locales del nodo para la ejecución de trabajos y tareas en el clúster.

```
Nodos/submit# yum -y install slurm slurm-devel slurm-perlapi slurm-torque slurm-contribs slurm-libs slurm-doc slurm-pmi slurm-pmi-devel   // Instalar slurmd
Nodos/submit# systemctl enable slurmd   // Habilitar slurmd
Nodos/submit# systemctl start slurmd   // Iniciar slurmd
Nodos/submit# systemctl status slurmd   // Status de slurmd
Nodos/submit# chown munge:munge /etc/munge/munge.key   // Cambiar usuario y grupo
```

4.1.3. Pruebas Slurm

En slurm se pueden usar dos tipos de comandos para mandar trabajos a slurm.

- **sbatch**: Se utiliza para enviar trabajos al sistema de gestión de trabajos Slurm para su ejecución programada.
- **srun**: Se utiliza para ejecutar tareas directamente en el clúster, generalmente de manera interactiva Proporciona una forma de ejecutar comandos y programas de forma inmediata y ver la salida en tiempo real.

Los comandos srun se ejecutan directamente en el cluster desde el nodo maestro. Usaremos los siguientes comandos de prueba de forma interactiva, ver imagen 43:

```
usuarioNormal$ sinfo   // Muestra información sobre el estado de los nodos
usuarioNormal$ srun[33] cal   // Muestra un calendario del mes actual en el terminal
usuarioNormal$ srun date   // Muestra la fecha y hora actual en el terminal
usuarioNormal$ srun hostname   // Muestra el nombre del host del nodo en el que se ejecuta el comando.

usuarioNormal$ srun sleep 5   // El nodo espere 5 segundos antes de finalizar.
usuarioNormal$ srun timedatectl   // Muestra información sobre la configuración de fecha y hora de uno de los nodos del cluster.
```

```
Maestro# scontrol update nodename=nodo1 state=idle // Permite modificar el estado del nodo 1
```

[33] srun: ejecuta tareas de forma interactiva directamente en los nodos de cómputo solicitados.

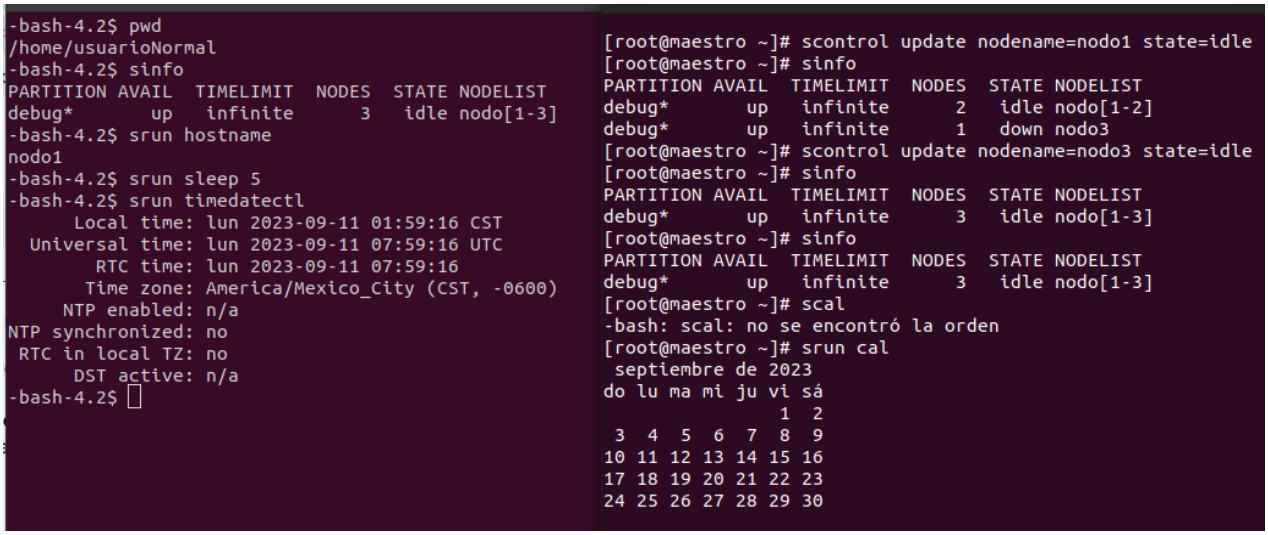

Imagen 43: Pruebas interactivas slurm en el cluster.

Para realizar las pruebas con sbatch utilizamos un usuario sin privilegios “usuarioNormal” el cual se encuentra en el cluster. Ahora generamos los archivos “**trabajo_prueba.sh**” y “**submit-hostname.sh**”.

“trabajo_prueba.sh”	“submit-hostname.sh”
#!/bin/bash VARIABLE="Ejemplo de programa" echo $VARIABLE echo "contando 20seg" sleep 20 cal	#!/bin/bash #SBATCH --job-name=test #SBATCH --output=res.txt #SBATCH --ntasks=2 #SBATCH --time=1:00 #SBATCH --mem-per-cpu=100 srun hostname srun sleep 20

Tabla 14: Archivos para realizar pruebas en slurm.

Para mandar los trabajos a slurm contamos con una serie de opciones que se muestran a continuación:
[nodomaestro]usuarioNormal**$** sbatch -p debug -n 4 -N 1 --cpus-per-task=1 trabajo_prueba.sh
Donde:

-p	// La partición que se define en el archivo slurm.conf
-n 4	// Número de cores que requiere el trabajo
-N 1	// Número de Nodos que requieren para el trabajo
--cpus-per-task	// CPUs por tarea

[nodomaestro]usuarioNormal$ sbatch submit-hostname.sh // Manda un trabajo al sistema slurm

El archivo submit-hostname genera un archivo de salida llamado “res.txt”. El cual se guardará en la carpeta de usuarioNormal (ver imagen 44).

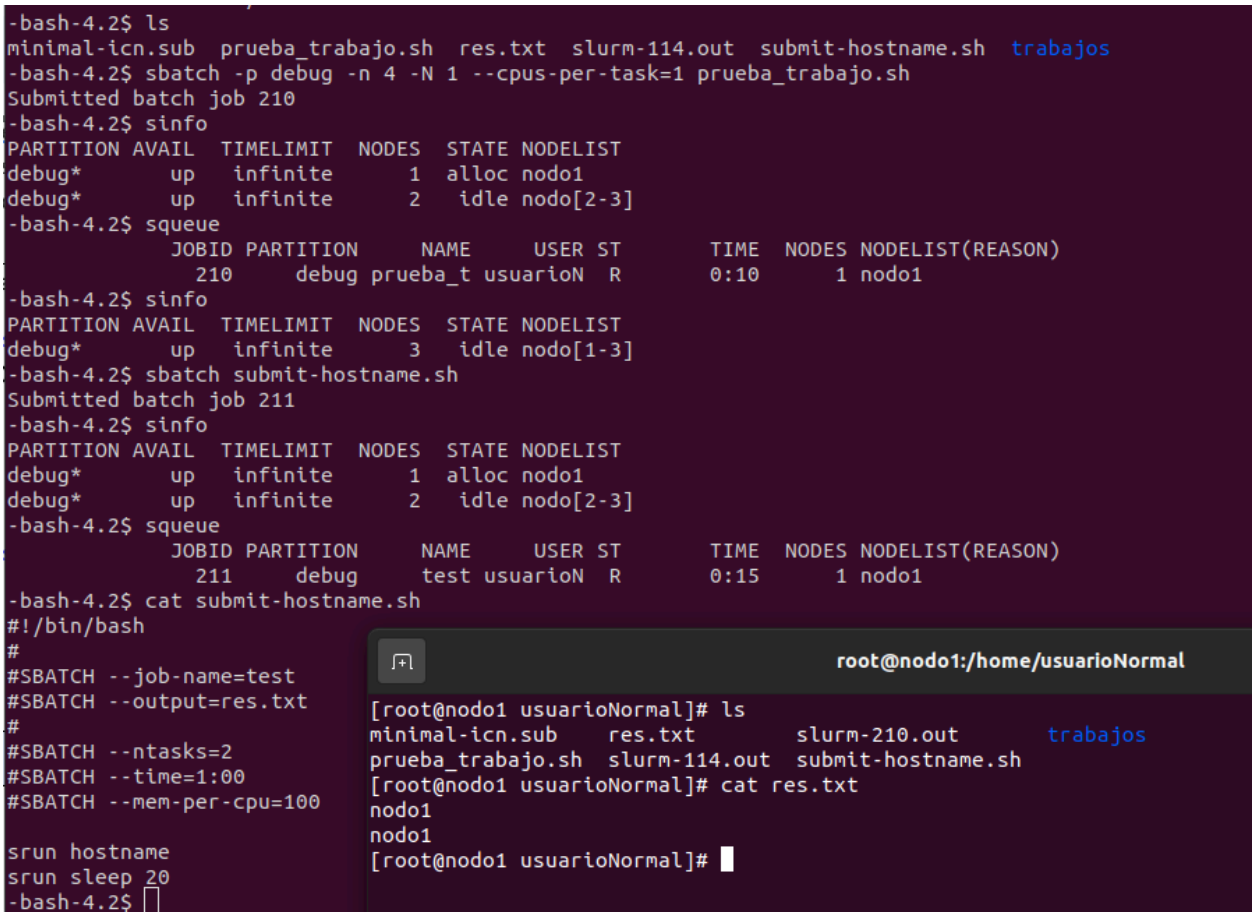

Imagen 44: Pruebas en slurm con el comando sbatch.

Una vez realizadas las pruebas:

- **Estados de los nodos (sinfo)**: Este comando nos permite verificar si los nodos están disponibles y configurados correctamente.
- **Ejecución de Trabajos (sbatch y srun):** Podemos concluir que Slurm funciona correctamente si los trabajos se ejecutan sin errores y producen los resultados esperados.
- **Monitoreo de Recursos (squeue):** Este comando verifica el estado de las colas y los trabajos en ejecución.

Podemos concluir que el cluster con slurm funciona correctamente. Estas comprobaciones corresponden al buen funcionamiento del componente 1 de nuestra maqueta. ver imagen 45.

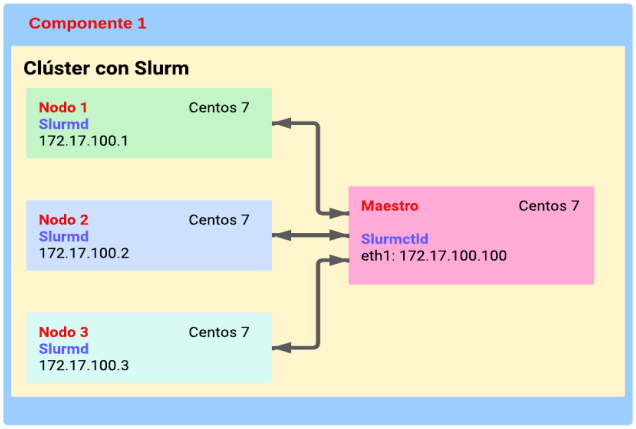

Imagen 45: Componente 1 de la maqueta.

Esta serie de configuraciones de nuestro Cluster con slurm se resumen en la tabla 15.

	Nodos	Configuraciones
Nodos Esclavos del cluster con slurm	Nodo 1 Nodo 2 Nodo 3	• Creación de llaves SSH, copiar llaves ssh a todos los nodos del cluster. • Configurar el tiempo en el servidor. • Modificar el archivo hosts para todos los nodos con sus IP correspondientes. • Crear una cuenta “usuarioNormal” sin privilegios. • Montar el directorio /home del maestro por NFS. • Instalar **epel-release.** • Instalación de **munge**, habilitar e iniciar el servicio **munge**. • Instalar **slurmd**, habilitar e iniciar el servicio slurmd. • Configuración del archivo **“etc/slurm/slurm.conf”** • Se utiliza el mismo archivo de configuración slurm en nodos esclavos.
Nodo externo al cluster, con la capacidad de enviar tareas al cluster mediante slurm	Submit	• Creación de llaves SSH, copiar llaves ssh a todos los nodos del cluster. • Configurar el tiempo en el servidor. • Modificar el archivo hosts para todos los nodos con sus IP correspondientes. • Crear una cuenta “usuarioNormal” sin privilegios. • Montar el directorio /home del maestro por NFS. • Instalar **epel-release.** • Instalación de **munge**, habilitar e iniciar el servicio **munge**. • Instalar slurm, habilitar e iniciar el servicio slurm. • Configuración del archivo **“etc/slurm/slurm.conf”**

Nodo maestro del cluster con slurm	Maestro	• Creación de llaves SSH, copiar llaves ssh a todos los nodos del cluster. • Configurar el tiempo en el servidor. • Modificar el archivo hosts para todos los nodos con sus IP correspondientes. • Crear una cuenta “usuarioNormal” sin privilegios. • Compartir el directorio /home por NFS a los nodos esclavos y submit. • Instalar **epel-release.** • Instalación de **munge**, habilitar e iniciar el servicio **munge**. • Creación de la llave munge y copiarla en los nodos esclavos . • Instalar slurmctld, habilitar e iniciar el servicio slurmctld. • Configuración del archivo **“etc/slurm/slurm.conf”** • Se utiliza el mismo archivo de configuración slurm del nodo maestro. • Pruebas Slurm.

Tabla 15: Configuraciones para obtener un cluster con slurm.

4.2. INDIGO IAM

Indigo Identity and Access Management (IAM) es una aplicación Spring Boot diseñada para ejecutarse detrás de un proxy inverso NGINX. Indigo IAM necesita cuatro requisitos previos para su funcionamiento:

- Este **proxy inverso** se encarga de descifrar el tráfico https entrante y luego lo reenvía a la aplicación IAM como tráfico HTTP (terminación TLS/SSL). El proxy inverso mejora el rendimiento
- Un **certificado X.509** que se utiliza para la terminación de SSL en el proxy inverso NGINX.
- Este servicio IAM mantiene el estado en una **base de datos** MariaDB/MySQL.
- Almacén de **claves JSON**. Este no es más que un archivo JSON Web Key (jwk). El cual representa una única clave pública o conjunto de llaves públicas, las cuales son utilizadas para firmar tokens. Dentro de iam permite utilizar el perfil wlcg[34] para emitir tokens.

Una vez cumplido con los requisitos previos es posible desplegar la aplicación INDIGO IAM en un contenedor.

[34] perfil wlcg: Se refiere a un perfil para generar tokens basado en el estándar JWT. Dicho perfil se desarrolló dentro de la infraestructura de la Worldwide LHC Computing Grid (WLCG).

4.2.1. Servidor NGINX Inverso

La instalación NGINX se realizó en el nodo iam. Este nodo tendrá la aplicación de INDIGO IAM. Una vez instalado se muestra el estatus del servicio, ver imagen 46.
Para la instalación de nginx se realiza mediante:
root@iam # apt install nginx // instalación de NGINX.
root@iam # systemctl status nginx // Ver el estatus del servicio NGINX.

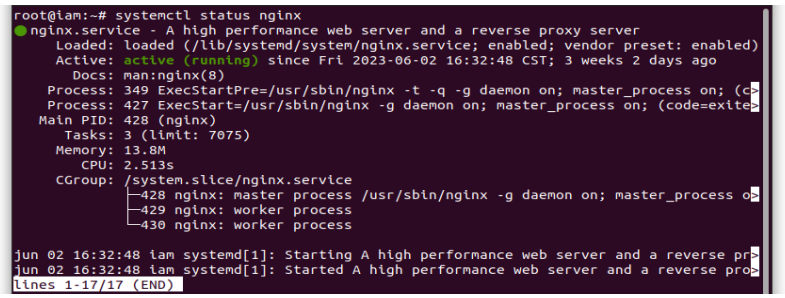

Imagen 46: Status del servicio nginx.

Una vez instalado el servidor nginx, se continúa con la configuración del proxy inverso. Al instalar el servidor se crea el directorio: ***/etc*/nginx/sites-available** en él existe un archivo llamado “default” el cual contiene la configuración del proxy por defecto.

El archivo “default” se renombra por “iam.atmosfera.unam.mx”. Ahora el archivo “iam.atmosfera.unam.mx” contendrá la configuración de nginx inverso. Dicha configuración se puede consultar en ver anexo de este documento.

root@iam# mkdir -p */var/*www/iam.atmosfera.unam.mx // Creacion del directorio contiene la página web que cargara nginx.

Después de tener el archivo iam.atmosfera.unam.mx se crea el link entre las carpetas.
root@iam# ln -s */etc*/nginx/sites-available/iam.atmosfera.unam.mx /etc/nginx/sites-enabled/
ya que el archivo de configuración del nginx inverso pertenece a ambas carpetas que hacen referencia a los sitios disponibles y habilitados.
Para comprobar el funcionamiento de nginx inverso se reinicia el servicio y se usa nginx -t el este comando comprueba la sintaxis del archivo de configuración.
root@iam # systemctl restart nginx
root@iam # nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful

root@iam # systemctl restart nginx // Verificamos el estado de nginx: activo e inverso

Ahora podemos ingresar al sitio: http://iam.atmosfera.unam.mx ya que nginx levantó el servicio ver imagen 47:

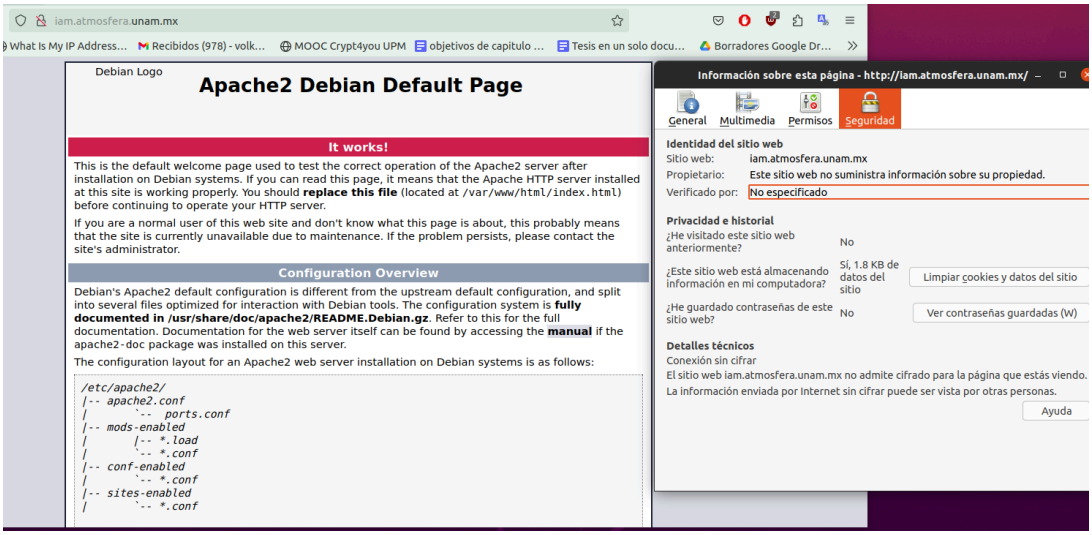

Imagen 47: Se muestra la página http://iam.atmosfera.unam.mx.

Como se mencionó anteriormente la carpeta /var/www/iam.atmosfera.unam.mx contiene la página web que carga nginx. Podemos crear un archivo de prueba y comprobar que nginx levanta la página en dicho directorio. Después de cargar de nuevo la configuración del servidor web podremos observar la página de prueba la cual nos indica que tenemos el servidor nginx funcionando. ver imagen 48:

```
root@iam # echo "HOLA prueba 1" > /var/www/iam.atmosfera.unam.mx/index.html //archivo index
root@iam # nginx -s reload            // Recargar la configuración de Nginx sin detener el servidor web
```

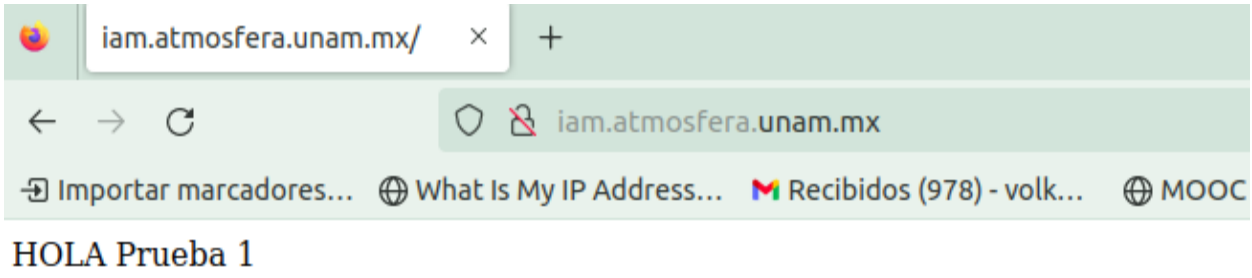

Imagen 48: Archivo index de prueba para Nginx.

4.2.2. Certificado del nodo iam

El siguiente paso es la obtención de un certificado. Let's Encrypt es una entidad y autoridad de certificación que ofrece certificados SSL/TLS de forma gratuita y automática. Para adquirir un certificado de Let's Encrypt para tu dominio, es necesario demostrar que tienes control sobre el mismo. En nuestro caso desde línea de comandos se utiliza el cliente ACME llamado Cerbot[35], que automatiza la emisión e instalación de certificados sin tiempo de inactividad.

```
root@iam # sudo apt install snapd                    // Instalar snapd[36]
root@iam # sudo snap install core; sudo snap refresh core // Instalar el paquete core de snap y actualizarlo
root@iam #  sudo snap install --classic certbot     // Instalar cerbot
root@iam#  sudo ln -s /snap/bin/certbot /usr/bin/certbot //Crear enlace simbolico en ambos directorios
root@iam # sudo cerbot --nginx                       // Obtiene un certificado nginx y los configura para activar el acceso https
```

Una vez que nos digan que se ha emitido el certificado para nuestro dominio revisamos el dominio y comprobamos que ahora nuestro dominio cuenta con https ver imagen 49.

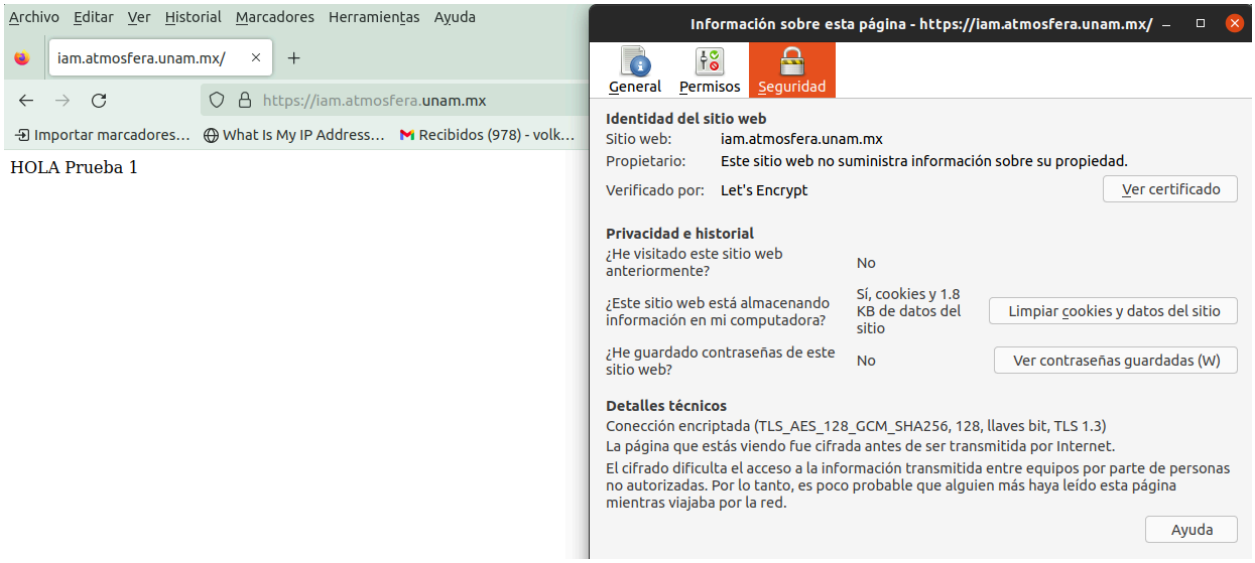

Imagen 49: Página con protocolo de transferencia de hipertexto seguro “HTTPS”.

[35] Certbot: Es un cliente de software desarrollado por la Electronic Frontier Foundation (EFF) que se utiliza para solicitar y gestionar certificados SSL/TLS de Let's Encrypt de manera automatizada. Está diseñado para simplificar el proceso de obtención, instalación y renovación de certificados, permitiendo a los usuarios implementar fácilmente HTTPS en sus sitios web.

[36] Snapd: Es el sistema de administración y ejecución de snaps en sistemas operativos basados en Linux

4.2.3. Base de Datos

El requisito de la base de datos se puede cumplir mediante una base de datos en el servidor o mediante un contenedor con MariaDB/MySQL. Puede resultar inseguro tener la base de datos en el servidor debido a esto usaremos podman[37] y crearemos un contenedor con Mysql. Este contenedor será utilizado por la aplicación INDIGO IAM.

```
root@iam# podman run --name iamBD -e MYSQL_ROOT_PASSWORD=#rootiam -p 3306:3306 -d mysql:latest  // Se crea el contenedor de nombre iamBD, con contraseña: #rootiam y se mapea al puerto 3306.

root@iam# podman exec -it iamBD mysql -uroot -p#rootiam  // Se ingresa al contenedor
```

Se creó una base de datos en el contenedor.

```
mysql>  CREATE DATABASE iamBD CHARACTER SET latin1 COLLATE latin1_swedish_ci;
```

Se creará el usuario con privilegios de administrador.

```
mysql>  CREATE USER 'admin'@'%' IDENTIFIED BY '#root#';
```

Privilegios de administrador al usuario recién creado y aplicar los cambios.

```
mysql>   GRANT ALL PRIVILEGES ON iamBD.* TO 'admin'@'%';
mysql>   FLUSH PRIVILEGES;
mysql>   exit
```

Usando el usuario root modificamos el método de autentificación para el usuario admin.

```
root@iam# podman exec -it iamBD mysql -uroot -p#rootiam
mysql>  ALTER USER 'admin'@'%' IDENTIFIED WITH 'mysql_native_password' BY '#root#'
mysql>   FLUSH PRIVILEGES;
mysql>   exit
```

Ahora, el usuario especificado debería estar configurado para utilizar el método de autenticación mysql_native_password en lugar de caching_sha2_password.

[37] Podman: es una herramienta de administración de contenedores que se utiliza en sistemas operativos basados en Linux

4.2.4. Claves JSON

Un almacén de Claves JSON se utiliza para contener *las claves empleadas en la firma de JSON web tokens.* En el contexto de INDIGO IAM (Identity and Access Management) estas *claves JSON desempeñan un papel esencial para firmar los tokens y verificarlos.*
Para crear una clave web JSON específica para su implementación en INDIGO IAM, se puede aprovechar la utilidad json-web-key-generator proporcionada por MitreID connect. Json-web-key-generator es una herramienta de línea de comandos basada en Java diseñada para generar JSON Web Keys (JWK) y JSON Private/Shared Keys (JPSK).

root@iam# git clone https://github.com/mitreid-connect/json-web-key-generator // bajar el git
root@iam#mvn package // maven construye el proyecto java en un formato específico en este caso jar.
A continuación, se generarán las llaves con el siguiente comando y se guardan en un archivo keystore.jwks : (ver imagen 50)
root@iam# java -jar target/json-web-key-generator-0.9-SNAPSHOT-jar-with-dependencies.jar \-t RSA -s **1024** -S -i rsa1 > keystore.jwks
Donde:

- **java**: Es el comando utilizado para ejecutar programas en la máquina virtual de Java (JVM).
- **-jar**: Indica a la JVM que se va a ejecutar un archivo JAR (Java Archive).
- **-t RSA**: Es un argumento que se pasa al programa que se ejecuta. En este caso, indica el tipo de algoritmo de generación de claves RSA.
- **-s 1024**: Es otro argumento que se pasa al programa y especifica el tamaño de la clave RSA que se generará, en este caso, 1024 bits.
- **arget/json-web-key-generator-0.9-SNAPSHOT-jar-with-dependencies.jar**: Es la ruta y el nombre del archivo JAR que se va a ejecutar.

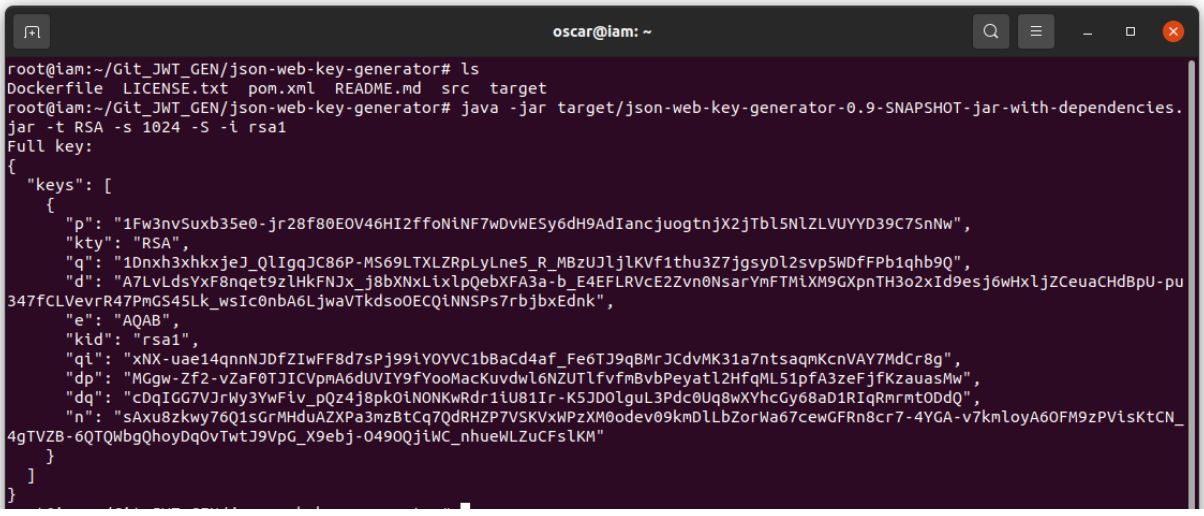

Imagen 50: Generación de llaves JSON.

4.2.5. Despliegue de la aplicación INDIGO IAM

Una vez reunidos los requisitos anteriores podemos hacer el despliegue de un contenedor por medio de podman, el cual contendrá nuestra aplicación de INDIGO IAM. Para esta maqueta usaremos la versión v1.8.2 de INDIGO IAM. La etiqueta de la imagen correspondiente a esta versión de la documentación es: indigoiam/iam-login-service:v1.8.2.

De acuerdo a la documentación [45] de INDIGO IAM necesitamos un archivo de configuración "*env*" el cual contiene la configuración de las variables de entorno para desplegar el contenedor con nuestra aplicación INDIGO IAM. Se puede consultar el archivo de configuración "env" en el anexo de este documento.

Previo al despliegue del contendor iam debemos crear una red y reubicar los archivos: "env", keystore.jwks en /home/userlocal/

root@iam# podman network create iam // Se crea una red personalizada en Podman, utilizada para conectar contenedores en el mismo host.

root@iam# mkdir /home/userlocal/ // creando el directorio.

root@iam# mv env keystore.jwks /home/userlocal // Moviendo archivos

Para desplegar el contenedor tenemos el siguiente comando ver imagen 51:

root@iam# podman run -it --name iam-login-service --net=iam -p 8080:8080 --env-file=/home/userlocal/env -v /home/userlocal/keystore.jwks:/keystore.jwks:ro --restart unless-stopped indigoiam/iam-login-service:v1.8.2

Donde:

- **podman run:** Esto inicia un nuevo contenedor utilizando Podman.
- **-it:** el contenedor se ejecuta en modo interactivo.
- **--name iam-login-service:** asigna un nombre al contenedor en este caso "iam-login-service".
- **--net=iam:** Esto especifica que el contenedor se conectará a una red llamada "iam".
- **-p 8080:8080:** mapea el puerto 8080 del contenedor al puerto 8080 del host.
- **--env-file=/home/userlocal/env:** Carga las variables de entorno usadas por el contendor.
- **-v /home/userlocal/keystore.jwks:/keystore.jwks:ro:** Monta el archivo keystore.jwks desde el host en el contenedor en modo solo lectura "ro".
- **--restart unless-stopped:** Esto especifica la política de reinicio del contenedor. En este caso, el contenedor se reiniciará automáticamente a menos que se detenga manualmente.
- **indigoiam/iam-login-service:v1.8.2:** Esta es la imagen de Docker que se utilizará para crear el contenedor.

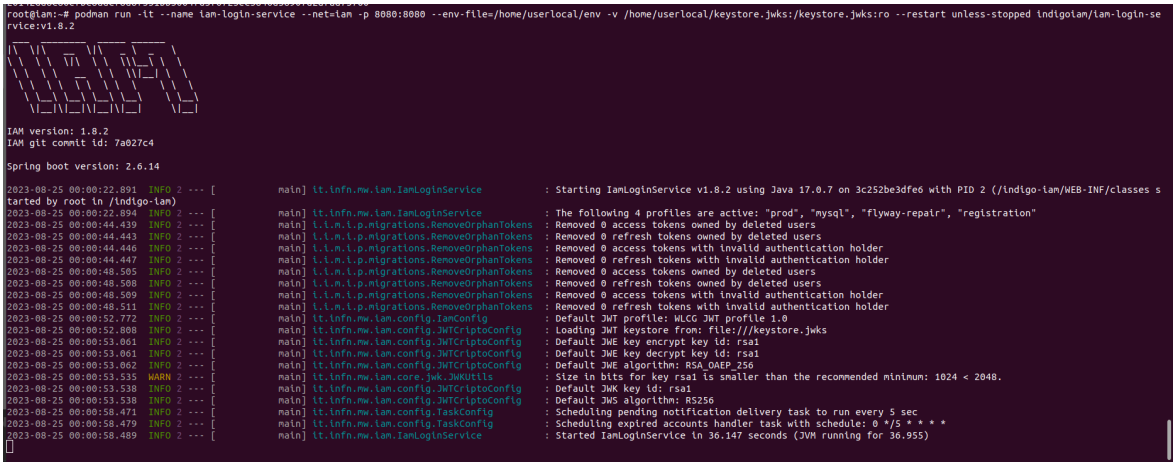

Imagen 51: Despliegue del contenedor INDIGO IAM.

Ya desplegado el contenedor es posible consultar la aplicación en https://iam.atmosfera.unam.mx/login ver imagen 52. Se cambian las credenciales por defecto y se ingresa a la aplicación .
Los usuarios pueden aplicar para una cuenta y obtener acceso a la plataforma una vez que dicha petición sea aprobada por el administrador.

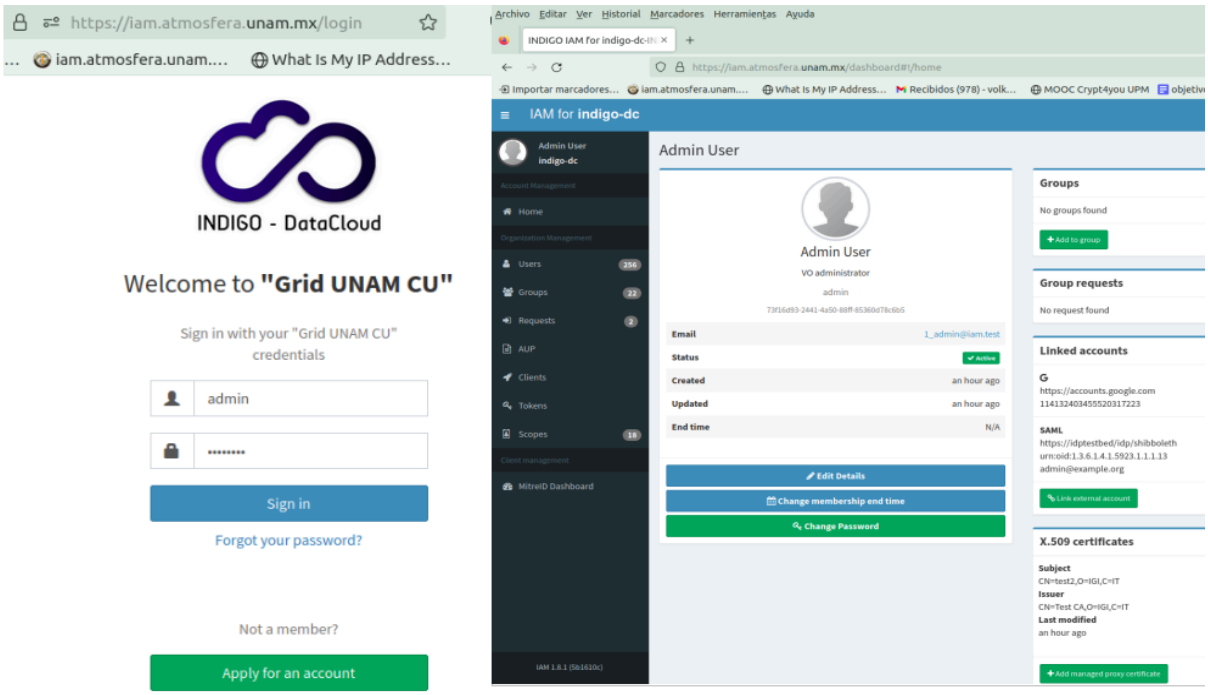

Imagen 52: Login y acceso a la aplicación INDIGO IAM como administrador.

Hasta este punto se han comprobado varias funciones como la base de datos con la información de usuarios, el despliegue de la aplicación y el uso del certificado ssl para https. ver imagen 53.

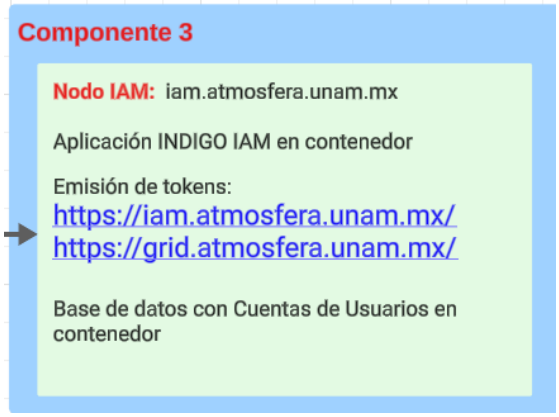

Imagen 53: Funcionalidades del nodo iam.

4.3. HTCondor-CE

HTCondor-CE es una configuración especial del software HTCondor diseñado como Compute Entrypoint (CE) para la computación en grid. Su función principal es delegar solicitudes de asignación de recursos (RAR) al sistema por lotes local, utilizando el demonio HTCondor Job Router. Esto permite que las organizaciones remotas envíen solicitudes para asignar temporalmente recursos informáticos locales.

Los sistemas por lotes compatibles con HTCondor-CE incluyen Grid Engine, HTCondor, LSF, PBS Pro/Torque y Slurm.

Este componente es quien se encargará de enviar los trabajos al sistema por lotes en nuestro caso slurm. Previo a su instalación se deben de cumplir ciertos requisitos como:

- Segunda interfaz con IP pública
- El nombre del servidor correspondiente al de la IP pública
- Certificado PKI para el servidor
- Repositorio EPEL instalado

HTCondor-CE debe instalarse en un host con la capacidad de presentar puestos de trabajo el clúster local. HTCondor-CE utiliza el certificado para encriptar el canal, hacer peticiones. Es importante Obtener el certificado mediante let's encrypt.

root@submit# yum clean all --enablerepo=* // limpia el cache de yum
root@submit# yum update // Actualización
root@submit# yum -y install fetch-crl // Instalar fetch-crl, Disponible en los repositorios de EPEL
El propósito principal de fetch-crl-cron es garantizar que las CRLs estén actualizadas y disponibles en el sistema para que los procesos y aplicaciones que verifican certificados digitales puedan verificar si un certificado en particular ha sido revocado o no.

root@submit# yum install fetch-crl-cron -y // Instalar fetch-crl-cron
root@submit# systemctl enable fetch-crl-cron // Habilitar fetch-crl-cron
root@submit# systemctl start fetch-crl-cron // Iniciar Fetch-crl-cron
root@submit# fetch-crl > /tmp/crl-$$.log 2>&1 < /dev/null &

Instalar repositorios[38] para poder instalar HTCondor-CE
root@submit# yum install https://research.cs.wisc.edu/htcondor/repo/10.0/htcondor-release-current.el7.noarch.rpm -y
root@submit# yum install htcondor-ce-slurm -y // Instalar HTCondor-CE

4.3.1. HTCondor-ce Sharing Spool

Al instalar HTCondor-CE se crea el directorio /var/lib/condor-ce el cual se debe compartir del nodo submit al nodo maestro y los nodos esclavos de nuestro clúster. Esto es debido a que los sistemas por lotes (System Batch) que no son HTCondor requieren un sistema de archivos compartido entre el host HTCondor-CE y los nodos de trabajo.
root@submit# cat /etc/exports // verificar el contenido del archivo exports

```
/var/lib/condor-ce 172.17.100.100/32(rw,no_subtree_check,sync,no_root_squash,async)
/var/lib/condor-ce 172.17.100.1/32(rw,no_subtree_check,sync,no_root_squash,async)
/var/lib/condor-ce 172.17.100.2/32(rw,no_subtree_check,sync,no_root_squash,async)
/var/lib/condor-ce 172.17.100.3/32(rw,no_subtree_check,sync,no_root_squash,async)
```

root@submit# systemctl enable nfs-server --now //Habilitamos el servicio NFS en el nodo submit
root@Maestro/Nodos# mkdir /var/lib/condor-ce //Creación del directorio en Nodos esclavos y maestro.

En los nodos esclavos y nodo maestro añadimos al archivo "/etc/fstab[39]" la siguiente línea:

```
submit:/var/lib/condor-ce /var/lib/condor-ce nfs defaults,_netdev 0 0
```

root@Maestro/Nodos# mount -a // Montamos manualmente el sistema de archivos del fstab
root@submit# systemctl enable htcondor-ce // Habilitar HTCondor-CE
root@submit# systemctl start htcondor-ce // Iniciar HTCondor-CE
root@submit# systemctl status htcondor-ce // Status HTCondor-CE

[38] Repositorios: Es el más actual a la fecha de realización de este trabajo, Condor Version: 10.0.9 2023-09-28
[39] fstab: Es un archivo de configuración (file systems table). Este archivo contiene información importante sobre los sistemas de archivos y particiones que deben montarse automáticamente durante el arranque del sistema.

4.3.2. Certificado para el nodo submit

El nodo submit debe de obtener su propio certificado tls para que encripte el canal para las peticiones a la Grid. Para esto se instala snapd, se activa el servicio y se siguen los pasos análogos previos a la instalación de cerbot en el nodo IAM. Al momento de obtener los certificados cerbot nos brinda dos opciones:

1. # sudo certbot --nginx: Ejecute este comando para obtener un certificado y haga que Certbot edite su configuración de nginx automáticamente para entregarlo, activando el acceso HTTPS en un solo paso.
2. # sudo certbot certonly --nginx: Ejecute este comando para obtener un certificado y realice los cambios en la configuración de nginx manualmente. Elegimos esta opción debido a que nos brinda el certificado con su respectiva llave. Estos archivos se colocan en una dirección específica para la configuración de HTCondor-CE. El certificado para autenticar conexiones SciToken y SSL.

A continuación se muestran los comandos utilizados para obtener el certificado ver imagen 53:

root@submit # sudo yum install snapd // Instalar snapd
root@ submit # systemctl enable –now snapd // iniciar snapd
root@submit # sudo snap install core; sudo snap refresh core // Instala y actualiza el paquete core de snap
root@submit# sudo ln -s /var/lib/snapd/snap /snap // enlace simbólico desde /var/lib/snapd/snap a /snap
root@submit # sudo snap install --classic certbot // Instalar cerbot
root@submit# sudo ln -s /snap/bin/certbot /usr/bin/certbot //Crear enlace simbolico en ambos directorios
root@submit # sudo yum install nginx // Instalar nginx
root@submit # sudo yum enable nginx // Habilita nginx
root@submit # sudo yum start nginx // Inicia nginx
root@submit#sudo certbot certonly --nginx // Obtenemos sólo el certificado.

Imagen 53: Obtención del certificado tls para el nodo submit.

Una vez obtenido el certificado este se renombra y se coloca de acuerdo a la documentación HTCondor-CE. HTCondor-ce utiliza el certificado y la llave para autenticar conexiones SSL y SciTokens. Estos certificados son necesarios para garantizar la autenticación y la seguridad de las comunicaciones. Existe un archivo: ca-bundle.crt[40] el cual sirve para que el servidor reconozca el certificado (submit.atmosfera.unam.mx) como auténtico. A continuación renombramos y colocamos el certificado según la documentación usando el método dos de la documentación [52]:

root@submit# cd /etc/letsencrypt/live/submit.atmosfera.unam.mx/ // Ubicación del certificado submit

root@submit# cp /etc/letsencrypt/live/submit.atmosfera.unam.mx/fullchain.pem /etc/pki/tls/certs/localhost.crt //Copiamos el certificado para el uso de htcondor-ce y cambiamos el nombre a localhost.crt

root@submit# openssl x509 -noout -subject -dates -in /etc/pki/tls/certs/localhost.crt // verificación del certificado del nodo submit, se muestran fechas de validez y el subject.

root@submit# cp privkey.pem /etc/pki/tls/private/localhost.key //Copiamos la llave para el uso de htcondor-ce y cambiamos de nombre.

Una vez instalado el certificado y la llave en */etc/pki/tls/certs/localhost.crt* y */etc/pki/tls/private/localhost.key* respectivamente. Cambiar los permisos y el propietario de la siguiente manera:

root@submit# chown root:root /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/localhost.key //Cambiamos el propietario y grupo a root.

root@submit# chmod 644 /etc/pki/tls/certs/localhost.crt // Cambia los permisos de lectura, escritura y ejecución

root@submit# chmod 600 /etc/pki/tls/private/localhost.key // Cambia los permisos de lectura, escritura y ejecución

[40] **ca-bundle.crt:** Es un archivo en CentOS 7 es un archivo que contiene certificados de autoridades de certificación confiables. Estos certificados se utilizan para verificar la autenticidad de certificados SSL/TLS al conectarse a sitios web seguros (HTTPS). El archivo se encuentra en /etc/pki/tls/certs/ca-bundle.crt y es crucial para que el sistema valide la autenticidad de los certificados.

4.3.3. Archivos de configuración en HTCondor-CE

Dentro de la configuración de HTCondor-CE existen varios archivos que se deben de configurar y adecuar para nuestro proyecto. dichos archivos son los siguientes:

1. "**/etc/condor-ce/mapfiles.d/50-gsi-callout.conf**": Este archivo hace uso de globus toolkit tecnología ya obsoleta y reemplazada por tokens.
2. "**/etc/condor-ce/config.d/02-ce-slurm.conf**": Este archivo de configuración ayuda a determinar la ruta de los sistemas batch, así como parámetros relacionados con el universo y trabajos. En nuestro caso el sistema batch es slurm.
3. "**/etc/blah.config**": Este archivo contiene las variables de los diferentes sistemas batch.
4. "**/etc/condor-ce/config.d/10-gridunam.conf**": Este archivo contiene los métodos de autenticación disponibles para la grid unam.
5. "**/etc/condor-ce/mapfiles.d/60-gridunam.conf**": Este archivo mapea los trabajos entrantes a la cuenta local "usuarioNormal" con el método de autenticación Scitokens.

Los archivos antes mencionados se configuran de la siguiente manera:

☑ El archivo de configuración "**/etc/condor-ce/mapfiles.d/50-gsi-callout.conf**" Comentar la linea:

```
#GSI /(.*)/ GSS_ASSIST_GRIDMAP
```

Debido a que globus Toolkit no se usará.

☑ El archivo de configuración: "**/etc/condor-ce/config.d/02-ce-slurm.conf**"
"HTCondor-CE Configuración del Sistema Batch" Este archivo establece la ruta para los trabajos en slurm, para esto indicamos cual es la cola predeterminada según la configuración de slurm (slurm.conf), en nuestro caso "debug". Las líneas deben quedar de la siguiente forma:

```
# Basic route for submitting to Slurm
JOB_ROUTER_ENTRIES @=jre
[
GridResource = "batch slurm";
TargetUniverse = 9;
name = "Local_Slurm";
set_default_queue = "debug";
]
@jre
```

☑ En el Archivo de configuración: "**/etc/blah.config"** Encontrar las siguientes variables y definir los siguientes valores:

```
supported_lrms=slurm,condor
blah_delegate_renewed_proxies=no
```

☑ El archivo de configuración "**/etc/condor-ce/config.d/10-gridunam.conf"** se crea y modifica con la siguiente información para especificar el método de autenticación disponibles para la grid unam. Los cuales dependiendo su orden de aparición es su orden de prioridad al momento del proceso de autenticación.

```
SEC_DEFAULT_AUTHENTICATION_METHODS = SCITOKENS,FS,PASSWORD
SEC_CLIENT_AUTHENTICATION_METHODS = SCITOKENS,FS,PASSWORD
SEC_CLIENT_AUTHENTICATION = REQUIRED
SEC_DEFAULT_AUTHENTICATION = REQUIRED

SCHEDD.SEC_READ_AUTHENTICATION_METHODS = $(SEC_DEFAULT_AUTHENTICATION_METHODS)
SCHEDD.SEC_WRITE_AUTHENTICATION_METHODS = $(SEC_DEFAULT_AUTHENTICATION_METHODS)

SCHEDD_DEBUG=D_FULLDEBUG,D_D_ALWAYS,D_SECURITY
```

☑ El archivo "**/etc/condor-ce/mapfiles.d/60-gridunam.conf"** permite mapear peticiones ya que una parte importante es contar con un usuario no root capaz de enviar trabajos al cluster. El usuario no root será utilizado para mapear los trabajos que llegan al nodo submit y se utilizarán los token emitidos por el servidor iam.atmosfera.unam.mx y grid.atmosfera.unam.mx.

HTCondor-CE Utiliza mapfiles unificados almacenados en **/etc/condor-ce/mapfiles.d/*.conf** para mapear los trabajos entrantes con credenciales en cuentas locales de Unix. Cada línea del archivo mapfiles consta de tres campos: Método de autenticación HTCondor, Credencial principal entrante formateado como Expresión regular compatible con PERL y una cuenta local.

Para permitir que los clientes con SciTokens ó WLCG envien solicitudes de trabajo a HTCondor-CE agregue lineas con el siguiente formato: SCITOKENS /<TOKEN ISSUER>,<TOKEN SUBJECT>/ <USERNAME>

Donde:

SCITOKENS es el método de autenticación.
<TOKEN ISSUER> es el emisor del token
<TOKEN SUBJECT> Es el sujeto al que se refiere el token.
<USERNAME> La cuenta local.

En nuestro caso el emisor del token será iam.atmosfera.unam.mx y gird.atmosfera.unam.mx, usamos la expresión regular (.*) para que acepte cualquier subject.

Las solicitudes serán mapeadas a la cuenta local “usuarioNormal” La siguiente configuración muestra el archivo mapfile: “**/etc/condor-ce/mapfiles.d/60-gridunam.conf”** con la siguiente configuración.

```
# SCITOKENS /<TOKEN ISSUER>,<TOKEN SUBJECT>/ <USERNAME>
SCITOKENS /^https:\/\/iam.atmosfera.unam.mx\/,(.*)/ usuarioNormal
SCITOKENS /^https:\/\/grid.atmosfera.unam.mx\/,(.*)/ usuarioNormal
SCITOKENS /^https:\/\/grid.atmosfera.unam.mx\/,0f2b7037-827e-55db-fc2a-f37e203e6444/ usuarioNormal
```

Una vez realizados los cambios anteriores se reinicia el servicio de condor-ce, se verifican los métodos de autenticación y se forza el uso de Scitokens como método de autenticación con los siguientes comandos desde el nodo submit.

root@submit# systemctl restart condor-ce.service // Reiniciar el servicio condor-ce

root@submit# condor_ce_config_val -v SCHEDD.SEC_READ_AUTHENTICATION_METHODS SCHEDD.SEC_WRITE_AUTHENTICATION_METHODS //Para verificar los métodos aceptados por el demonio SCHEDD con el comando condor_ce_config_val

root@submit# export _condor_SEC_CLIENT_AUTHENTICATION_METHODS=SCITOKENS // Para forzar el uso de Scitokens para solicitudes a condor-ce se exporta la variable con el valor de scitokens.

root@submit# export _condor_TOOL_DEBUG=D_FULLDEBUG,D_D_ALWAYS,D_SECURITY // Define una variable para ayudarnos a determinar problemas al momento de usar condor_ce_ping

4.4. OpenID Connect

OpenID Connect (OIDC) es un protocolo de autenticación basado en OAuth 2.0. que se utiliza en la autenticación segura de usuarios y la transmisión de información de perfil de usuario entre proveedores de identidad como Google, Microsoft, IBM, Azure y otros.
OIDC permite a los usuarios autenticarse en aplicaciones utilizando cuentas existentes en proveedores de identidad. En nuestro nodo submit usamos OIDC-Agent el cual es un conjunto de herramientas diseñadas para gestionar tokens desde línea de comandos. Este agente permite a los usuarios manejar y utilizar estos tokens de manera segura y eficiente. Oidc-agent generalmente se inicia al principio de una sesión X. S*e debe de iniciar el agente y cargar el perfil de usuario* si se cuenta con él ó se crea una configuración de usuario nueva.

4.4.1. Cuenta INDIGO-IAM

Un paso importante es contar con una cuenta existente en INDIGO. Para esto es necesario ingresar a https://grid.atmosfera.unam.mx/login ó https://iam.atmosfera.unam.mx/login. Una vez se ingresa a la página es necesario aplicar para una cuenta, llenar el formulario y enviar la petición. La cual *deberá ser autorizada por el administrador* y se enviará por correo una liga para establecer una contraseña (el correo es enviado dentro del dominio *unam.mx en la bandeja de spam) ver imagen 54.

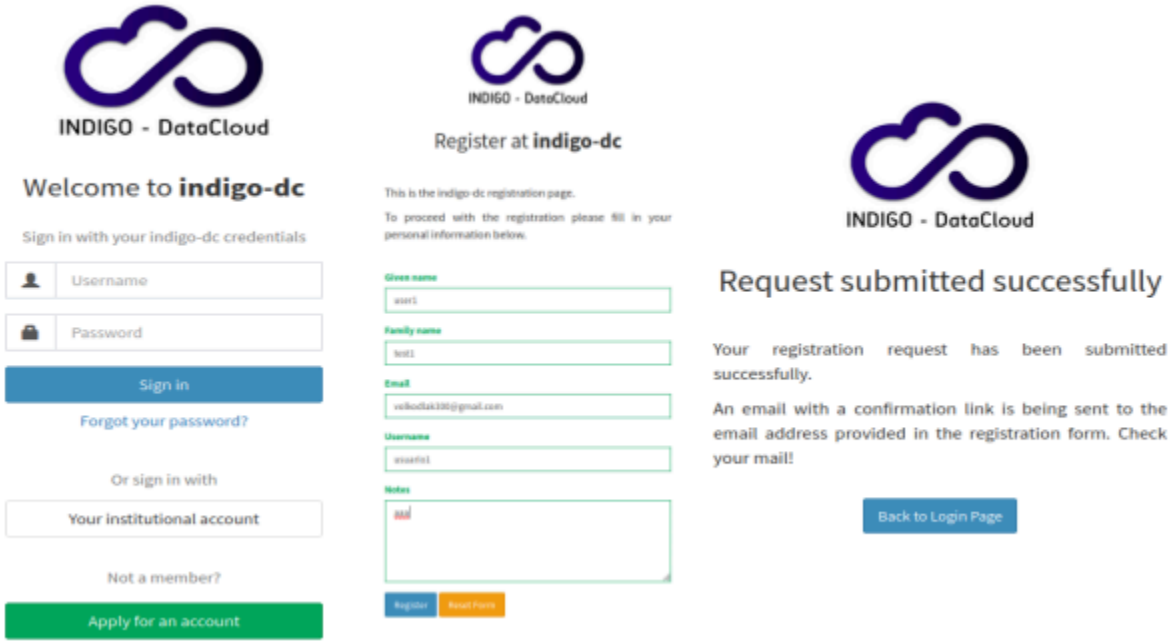

Imagen 54: Proceso de aplicación a una nueva cuenta de usuario a la Grid.

4.4.2. Instalación y Configuración OIDC

Una vez que se tiene una cuenta de INDIGO se procede a instalar oidc y se realiza el registro del perfil OIDC. Este registro de usuario es el proceso de autenticación, donde el propietario de la cuenta INDIGO-IAM otorga permiso a la aplicación OpenID Connect, para acceder a su cuenta y otorgar permisos a INDIGO para usar información del perfil y permitir a la aplicación peticiones de token en nombre del usuario dueño de la cuenta INDIGO-IAM. La versión más actual es oidc-5.1.0-1.el7 nosotros trabajamos con la versión oidc-4.5.2.1 debido a que grid.atmosfera.unam.mx trabaja con la version 4.5.2.1 y existe un problema de compatibilidad entre las versiones 4 y 5. Por esta razón usamos versionlock.

```
root@submit # yum -y install https://repo.data.kit.edu/centos/7/oidc-agent-4.5.2-1.el7.x86_64.rpm
//Instalación de oidc desde repositorio.
root@submit# sudo yum install yum-plugin-versionlock // software para bloquear la versiones
root@submit# sudo yum versionlock add oidc-agent // Se bloquea oidc
```

4.4.2.1. Cuenta de configuración para usuarios

El comando oidc-gen permite crear una configuración de cuenta que será usada por el cliente para obtener tokens de INDIGO. La obtención de tokens entre OIDC-Agent e INDIGO-IAM implica el uso del diagrama de flujo conocido como "Device Authorization Flow" para el registro OpenID profile ver imagen 55. Recordemos que Indigo Identity and Access Management (INDIGO-IAM) es una herramienta de software que proporciona la emisión de token conforme al perfil Worldwide LHC Computing Grid (WLCG), para la solicitar un token se requiere tener una cuenta en INDIGO.

El procedimiento es el siguiente:
Lo primero que debe hacer es iniciar oidc-agent, desde un usuario sin permisos. Esto se puede hacer emitiendo el siguiente comando:
root@submit # sudo useradd -u 1006 -m -c "prueba4" -d /home/prueba -s /bin/bash prueba4
root@submit # su -l prueba4 // Se ingresa a una cuenta sin permisos root “prueba4”.
prueba4@submit $ eval $(oidc-agent) // Se inicia el agente oidc
prueba4@submit$ oidc-gen -w device prueba1 --issuer=https://grid.atmosfera.unam.mx/ --scope="condor:/READ condor:/WRITE" //Se realiza la petición oidc-gen la cual genera una nueva configuración, -w indica el método de registro en este caso “device” y el nombre del perfil “prueba1”. --issuer, es la url del emisor de token en este caso https://grid.atmosfera.unam.mx/ la ultima diagonal “/” es importante. --scope, establece los alcances de los tokens que serán emitidos.

De manera inmediata el servidor proporciona un código que aparece en línea de comandos, el cual debe ser ingresado a la página web: https://grid.atmosfera.unam.mx/device En la cual debemos ingresar nuestras credenciales. Una vez hecho esto, podremos acceder a la siguiente página para poder ingresar el código que se nos generó en la terminal, ver imagen 56.

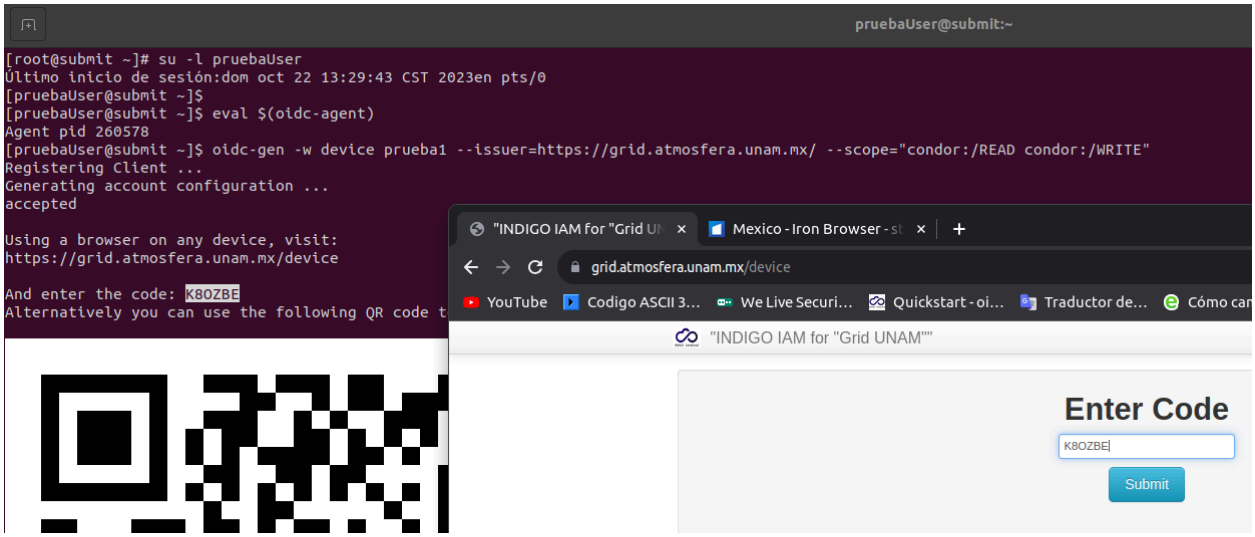

Imagen 56: Proceso de generación de una nueva cuenta de configuración

Una vez que se ingresa el código y es correcto (autenticación): Nos mostrará la siguiente página https://grid.atmosfera.unam.mx/device/verify Donde nos indica que fue el cliente registrado dinámicamente y que fue aprobado. Ahora autorizamos (autorización) a oidc-agent:prueba1 para acceder a nuestra información, así como los permisos que tendrán los tokens para escritura y lectura. De esta manera se ha registrado el perfil, ver imagen 57.

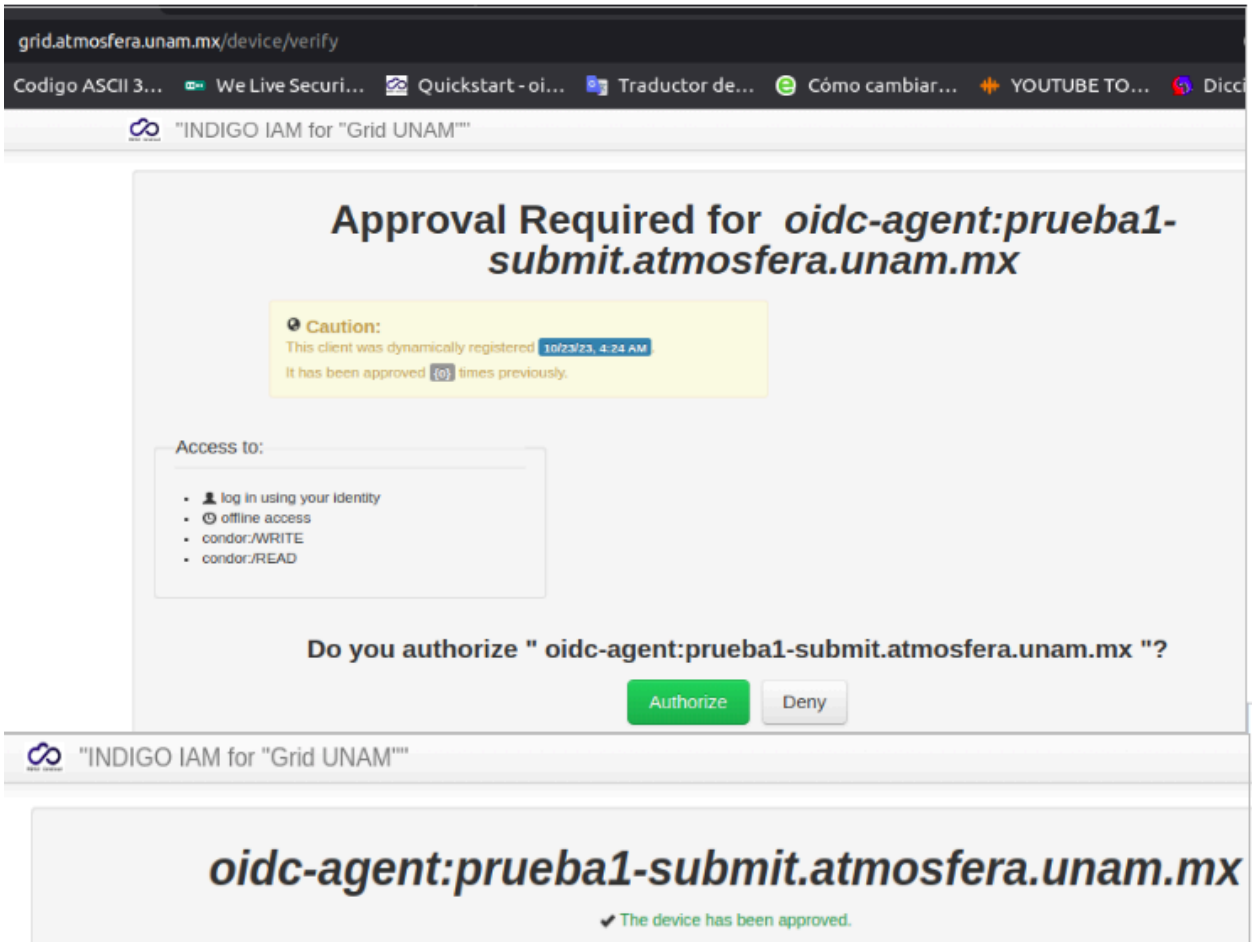

Imagen 57: Proceso de Autorización para la cuenta de configuración prueba1.

Si el registro es exitoso se solicitará en terminal una contraseña para encriptar el perfil ”prueba1” y pueda ser reutilizado para futuras solicitudes de token, ver imagen 58.

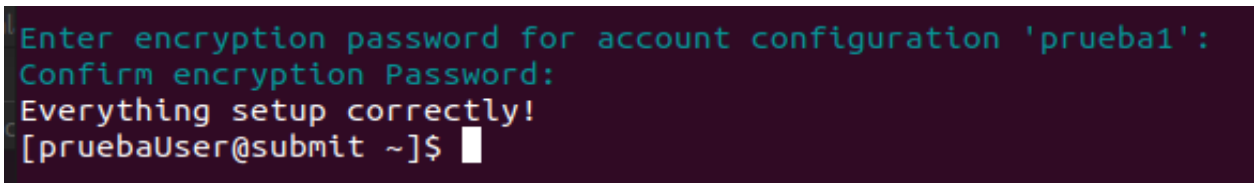

Imagen 58: Registro exitoso para la cuenta prueba1.

El proceso antes mencionado está basado en “Device Authorization Flow” este permite a clientes OAuth en dispositivos con limitaciones de entrada, obtengan la autorización del usuario para acceder a recursos protegidos (cuenta INDIGO). En este proceso de autorización se instruye al usuario para revisar la solicitud de autorización en un dispositivo secundario, el cual cuenta con la capacidad de entrada y un navegador adecuado para conectarse al servidor de autorización para aprobar la solicitud de acceso.

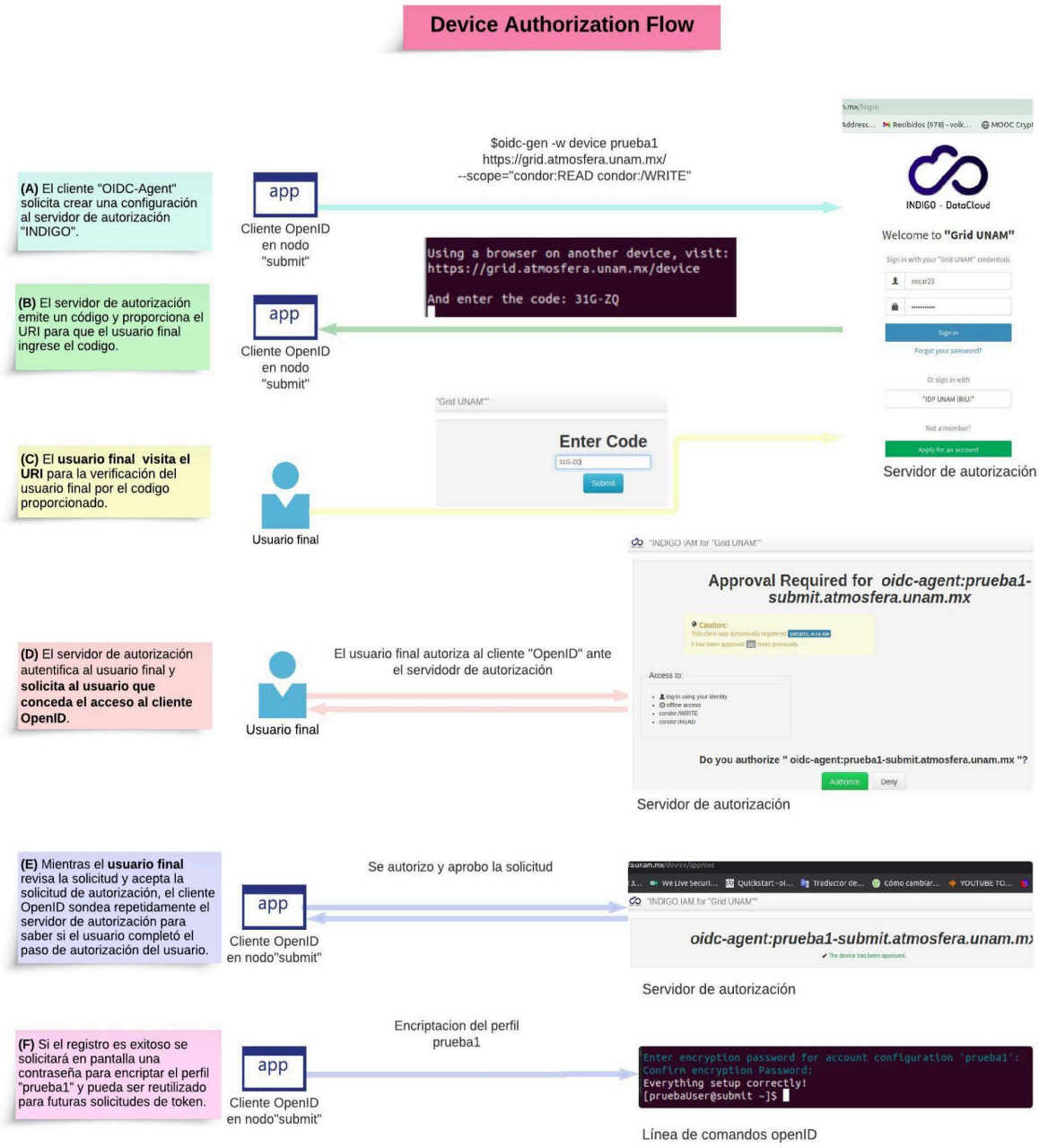

Imagen 55: Diagrama “Device Authorization Flow”.

5. Verificación completa de la maqueta

El propósito de este trabajo radica en la autenticación de usuarios por medio de tokens, dichos tokens son emitidos por la grid unam. Después dicho token es utilizado para su autenticación y uso al momento de enviar trabajos al cluster desde el nodo submit, ver imagen 59.

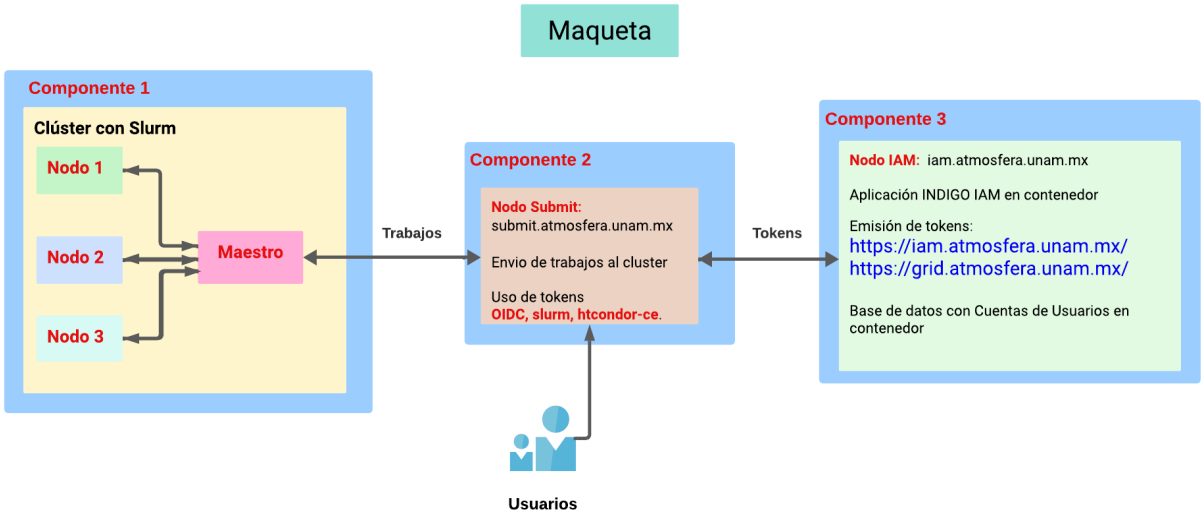

Imagen 59: Maqueta dividida en componentes y sus funciones.

Por lo anterior las pruebas serán divididas en dos secciones, la autenticación de usuarios por medio de los scitokens y el envío de trabajos htcondor utilizando los scitokens. El hecho de enviar trabajos muestra el completo uso y éxito de nuestra maqueta.
Un paso previo a las pruebas es obtener un token que será utilizado para autenticar el usuario y para mandar trabajos al cluster.
Para realizar las pruebas necesitamos obtener el token de grid unam. Previo a esto necesitamos contar con los pasos previos del punto 4.4.4.1 “Cuenta de configuracion para usuarios”. ya realizados. Una vez con este punto hecho se realizan los siguientes pasos para obtener el token para la autenticación:

prueba4@submit $ eval $(oidc-agent) // Se inicia el agente oidc
prueba4@submit $ oidc-add -l // Se listan las cuentas oidc configuradas
prueba4@submit $ oidc-add prueba1 // Se elige la cuenta prueba1
prueba4@submit $ oidc-token --aud="submit.atmosfera.unam.mx:9619" prueba1 // Se lleva a cabo una solicitud de un token con la herramienta oidc-token indicando el perfil generado previamente con la herramienta oidc-gen. En este caso la cuenta usa es “prueba1” y debe indicarse el campo de audiencia “aud” que debe contener el token y no es otra cosa que el nombre del servidor al cual se solicitaron los recursos seguido del puerto usado por HTCondor “9619”. Ver imagen 60.
Para solicitar recursos al sistema condor-ce se debe proporcionar el token correspondiente por medio de la variable de entorno BEARER_TOKEN o en un archivo cuya ruta se debe especificar en la variable BEARER_TOKEN_FILE.

Para almacenar el token en la variable BEARER_TOKEN se puede utilizar el siguiente comando:
prueba4@submit$export BEARER_TOKEN=$(oidc-token --aud=submit.atmosfera.unam.mx:9619 prueba1)
prueba4@submit$ export _condor_SEC_CLIENT_AUTHENTICATION_METHODS=SCITOKENS // Se forza el método de autenticación para usar scitokens
prueba4@submit$ export _condor_TOOL_DEBUG=D_FULLDEBUG,D_D_ALWAYS,D_SECURITY // Se define la variable debug que es parte de las herramientas de condor para verificar errores

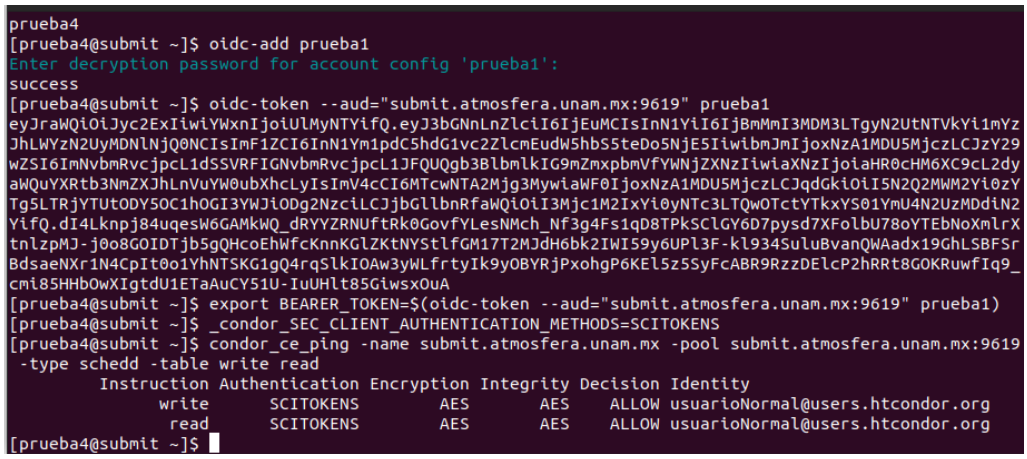

Imagen 60: Obtención y uso de un token por medio de OIDC.

Para verificar información del token se puede usar https://jwt.io/ o el programa en bash“verToken.sh”. Ver imagen 61. El código se encuentra en el anexo de este trabajo.

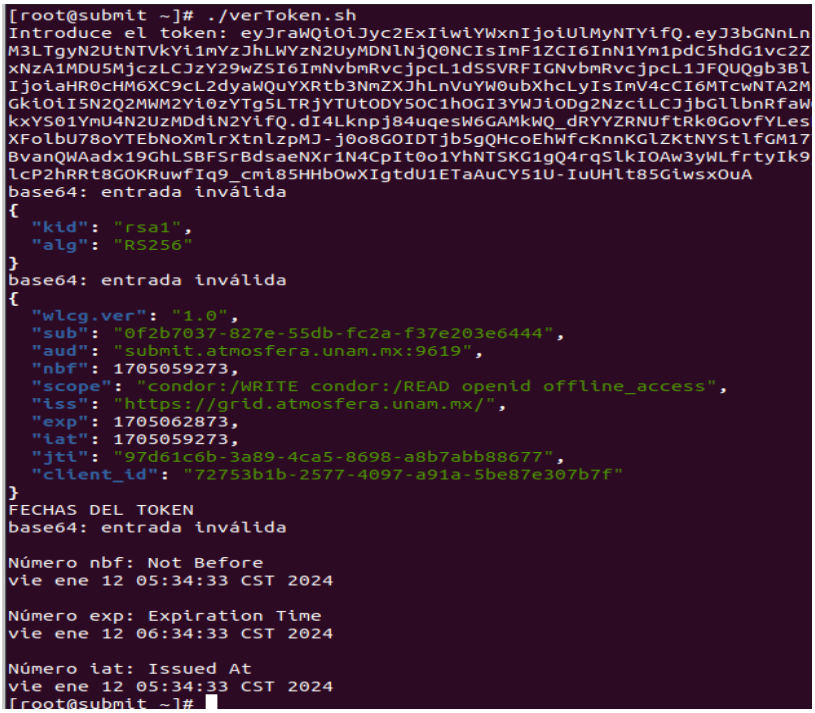

Imagen 61: Decodificación de un token por el programa verToken.sh.

5.1. Pruebas de Configuración y Funcionalidad en HTCondor-CE

Ahora que contamos con un Token funcional se realizan pruebas a la maqueta que nos permiten determinar su correcto funcionamiento. Se verifica el estado de HTCondor-ce, la red, la capacidad de enviar trabajos, el proceso de envío de trabajos, enviar trabajos simples y listar dichos trabajos. Las pruebas son las siguientes:

- ☑ Prueba de estatus: *condor_ce_status*.
- ☑ Prueba de Red: *condor_ce_host_network_check*.
- ☑ Prueba de capacidad de enviar trabajos: *condor_ce_ping*.
- ☑ Prueba de funcionamiento del HTCondor-CE: *condor_ce_trace*.
- ☑ Ejecución de un proceso en el nodo HTCondor-CE (submit) indicado: *condor_ce_run*.
- ☑ Ejecución de un proceso en un nodo del cluster: *condor_ce_run*.

5.1.1. Prueba de estatus: *condor_ce_status*

El comando **condor_ce_status** nos ayuda a ver los demonios ejecutándose en el entorno HTCondor-CE. Esto nos ofrece algunas pautas para diagnosticar problemas, por ejemplo al realizar el **condor_ce_status -any** y no muestra al menos los siguientes demonios: Collector, Scheduler, DaemonMaster, Job_Router.. Podría indicar un problema con el funcionamiento del HTCondor-CE, ver imagen 62.
root@submit# condor_ce_status -any

```
[root@submit private]# condor_ce_status -any
MyType              TargetType          Name

Collector           None                My Pool - submit.atmosfera.unam.mx@submit
Grid                None                batch
Job_Router          None                htcondor-ce@submit.atmosfera.unam.mx
Submitter           None                prueba4@users.htcondor.org
Submitter           None                prueba@users.htcondor.org
Scheduler           None                submit.atmosfera.unam.mx
DaemonMaster        None                submit.atmosfera.unam.mx
Submitter           None                usuarioNormal@users.htcondor.org
```

Imagen 62: Prueba de estatus: *condor_ce_status.*

5.1.2. Prueba de Red: *condor_ce_host_network_check*.

El comando condor_ce_host_network_check es una herramienta que nos permite verificar la configuración de red de un HTCondor-CE. Esta herramienta nos brinda el nombre del host, el nombre del dominio completo el cual coincide en nuestro caso con el host. Verifica la configuración del DNS. Nos dice si nuestra configuración de red es adecuada, ver imagen 63.
root@submit# condor_ce_host_network_check

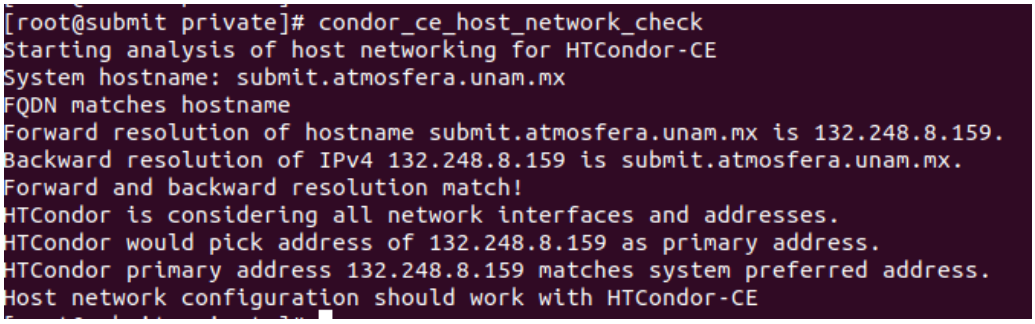

Imagen 63: Prueba de Red: condor_ce_host_network_check.

5.1.3. Prueba de capacidad para enviar trabajos: *condor_ce_ping*.

El comando **condor_ce_ping** es utilizado para probar la capacidad para enviar trabajos a un HTCondor-CE. Su objetivo es verificar la capacidad de envío de trabajos al CE y obtener información sobre la autorización, métodos de autenticación y otras configuraciones relacionadas.

prueba4@submit$ condor_ce_ping -name submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 -type schedd -table write read

El comando **condor_ce_ping** permite verificar los permisos de escritura/lectura así como el tipo de autenticación usado. En nuestro caso se utiliza el método de autenticación Scitokens ver imagen 64.

Esta herramienta permite ver el algoritmo de encriptación y si se aprueba la petición, también podemos ver el "usuarioNormal" el cual es un usuario existente dentro del cluster sobre el cual se mapean las peticiones de trabajos, de acuerdo al archivo de configuración mapfiles ubicado en "/etc/condor-ce/mapfiles.d/60-gridunam.conf" correspondiente al software de htcondor-ce.

El comando condor_ce_ping cuenta con la opción -table la cual nos indica el formato en que nos presentará la información relacionada con los permisos de lectura y escritura.

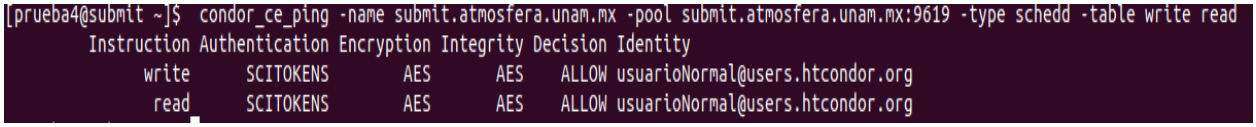

Imagen 64: Prueba de capacidad para enviar trabajos: *condor_ce_ping*. Autenticación con SCITOKENS

Este segundo comando con condor_ce_ping es como el anterior pero con la opción -verbose reemplazando la opción table.

prueba4@submit$ condor_ce_ping -name submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 -type schedd -verbose write read

Verbose nos brinda más información sobre el método de autenticación usado, los métodos de autenticación disponibles, los permisos de lectura/escritura, el tipo de encriptación, la versión de htcondor entre otros, ver imagen 65.

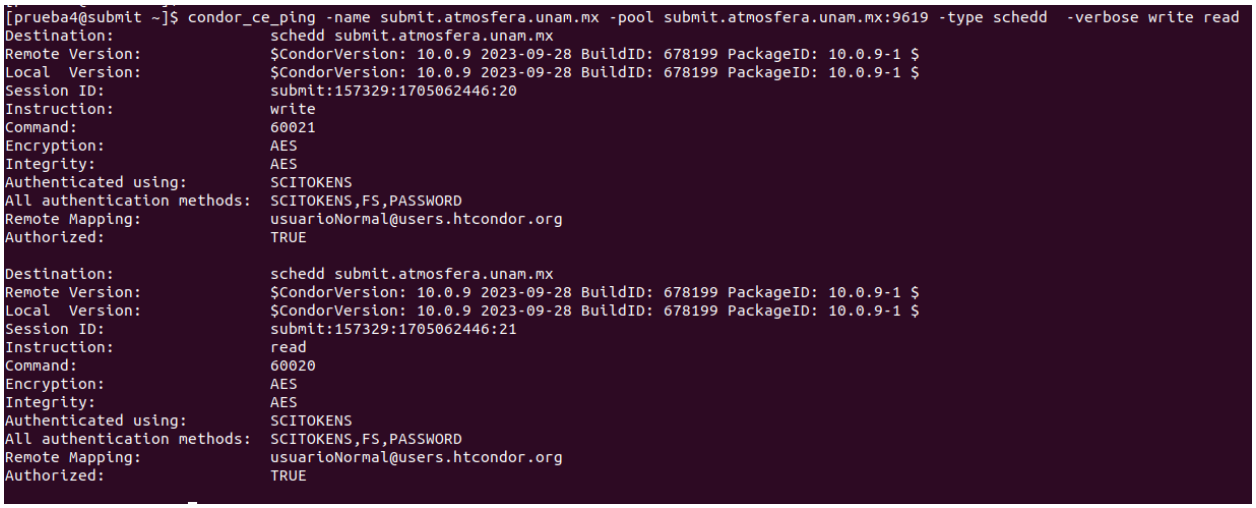

Imagen 65: Prueba de capacidad para enviar trabajos: *condor_ce_ping*. Autenticación con SCITOKENS desglosado.

5.1.4. Prueba de funcionamiento del HTCondor-CE: *condor_ce_trace*

El comando **condor_ce_trace** es una herramienta útil para probar el envío de trabajos de extremo a extremo en un entorno HTCondor-CE. Su función principal es verificar el proceso completo de envío de trabajos, desde la verificación de permisos para enviar trabajos al CE hasta el seguimiento del trabajo resultante.

prueba4@submit$ condor_ce_trace --debug submit.atmosfera.unam.mx

El comando anterior con la opción --debug nos muestra en consola: Los archivos de configuración mapfiles.d que son leídos para establecer la autenticación por scitokens, nos muestra el certificado público del servidor, ver imagen 66. También una lista de configuraciones del propio htcondor-ce. En otras palabras este comando nos muestra la traza de los proceso ocurrentes al momento de enviar un trabajo, ver imagen 66.

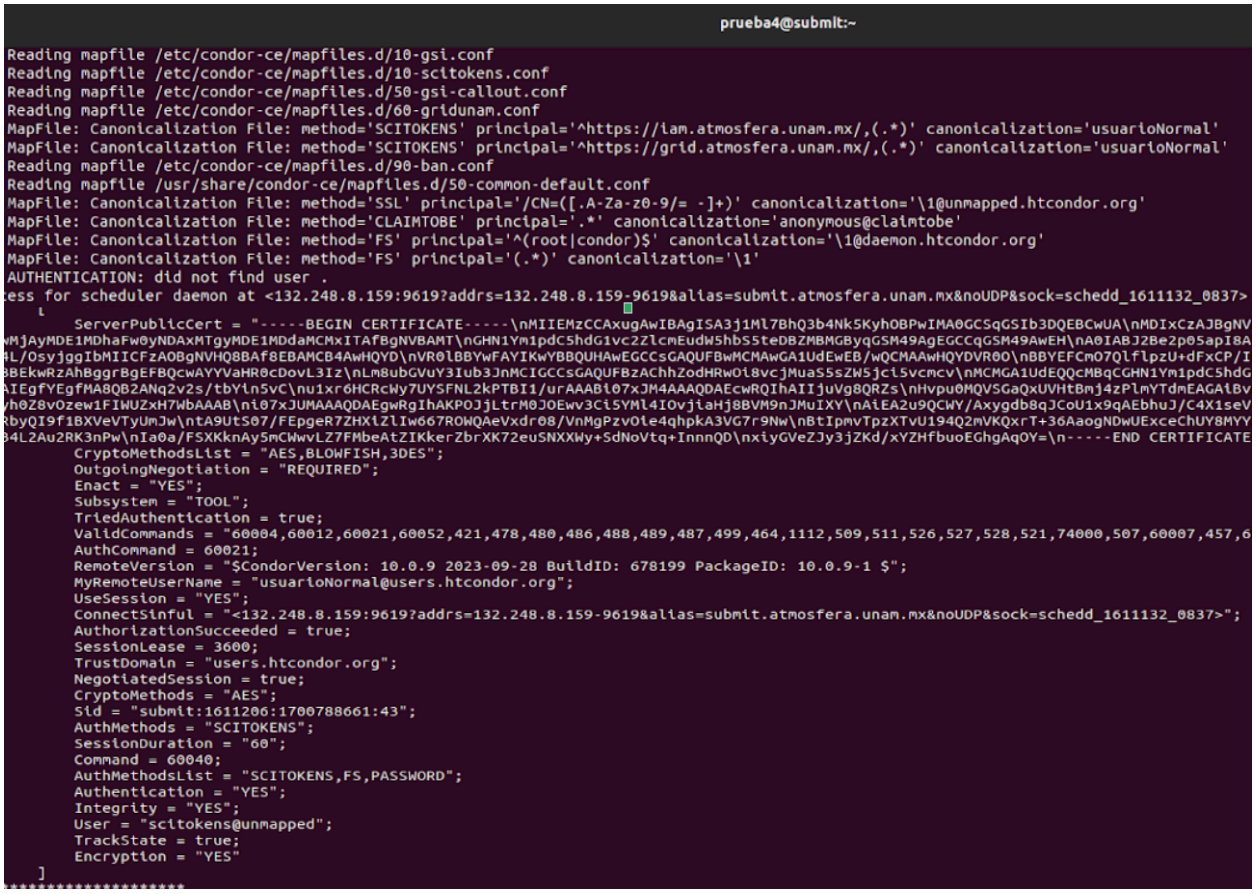

Imagen 66: Prueba de funcionamiento del HTCondor-CE: *condor_ce_trace*.

5.1.5. Ejecución de un proceso en el nodo HTCondor-CE (submit) indicado: *condor_ce_run*.

El comando **condor_ce_run** es una herramienta que envía un trabajo simple al CE. Es una herramienta útil para enviar trabajos de forma rápida y sencilla a través del CE. Así como la ejecución de comandos en el sistema batch remoto.

Las opciones de esta prueba son:

-l, --local: Esta opción indica que el trabajo debe ejecutarse en el universo local del CE. Cuando se utiliza, el trabajo se ejecuta en la máquina donde se encuentra el CE, sin ser enviado a un sistema batch remoto.

-r, --remote: Esta opción indica que el trabajo se enviará directamente al sistema batch remoto asociado con el CE, saltando el proceso local. Permite enviar un trabajo al CE para que se ejecute en el sistema batch remoto.

prueba4@submit$ condor_ce_run -lr submit.atmosfera.unam.mx:9619 /bin/hostname
Este comando especifica que se ejecute de manera local el comando "/bin/hostname"

En nuestra prueba el nodo que responde al comando es submit.atmosfera.unam.mx, ver imagen 67. El mismo nodo que ejecuta el comando.

```
[prueba4@submit ~]$ condor_ce_run -lr submit.atmosfera.unam.mx:9619 /bin/hostname
submit.atmosfera.unam.mx
```

Imagen 67: Prueba condor_ce_run local.

5.1.6. Ejecución de un proceso en un nodo del cluster: *condor_ce_run*.

Ahora se utiliza condor_ce_run de manera remota:

prueba4@submit$ condor_ce_run -r submit.atmosfera.unam.mx:9619 /bin/hostname

Este comando especifica que se ejecute de manera remota el comando "/bin/hostname" después de usar el comando, el nodo que respondió fue el nodo1. El cual es un nodo perteneciente a nuestro cluster, ver imagen 68.

```
[prueba4@submit ~]$ condor_ce_run -r submit.atmosfera.unam.mx:9619 /bin/hostname
nodo1
```

Imagen 68: Prueba condor-ce run remota.

5.2. Pruebas de Envío de Trabajos HTCondor-ce

En este apartado exploramos diferentes escenarios de envío de trabajos HTCondor a la grid mediante el uso de tokens. Los trabajos abarcan desde tareas simples hasta flujos de trabajo más complejos. [51]
Para el envío de trabajos se cuenta con un archivo descriptivo que especifica los recursos de memoria, procesador, almacenamiento, así como el programa a ejecutar, el universo[41], etc. Las pruebas de envió de trabajo se muestran de forma incremental respecto a su complejidad. Las pruebas son:

- ☑ Prueba de Envío del trabajo 1: "submit.sub"
- ☑ Prueba de Envío del trabajo 2: "Test-script.sh"
- ☑ Prueba de Envío del trabajo 3: "programa_hola_mundo.sh"
- ☑ Prueba de Envío del trabajo 4: "ejercicio_simple_en_C.c"
- ☑ Prueba de Envío del trabajo 5: "calculaPI_Instancias.c"
- ☑ Prueba de Envío del trabajo 6: "wordcount-top-n.py"

5.2.1. Prueba de Envío del trabajo 1: "submit.sub"

La primera prueba consta de enviar un comando sencillo en shell, presentando en el archivo descriptor del trabajo. Este comando sencillo es /bin/hostname.

Trabajo 1 "submit.sub"	
Programa "hostname.sub"	Descripción
executable = /bin/hostname **output** = hostname.out **error** = hostname.err **log** = hostname.log **request_cpus** = 1 **request_memory** = 1MB **request_disk** = 1MB queue	**executable**: Es el nombre del programa que se ejecutará. **output**: Indica el nombre del archivo donde HTCondor escribirá la salida estándar del trabajo. **error**: Especifica el nombre del archivo donde HTCondor escribirá el error estándar del trabajo. **log**: Es el nombre del archivo donde HTCondor escribirá información sobre el registro del trabajo. **request_***: Son los requerimientos específicos del programa que HTCondor-CE debe considerar al ejecutarlo. **queue**: Es una instrucción que indica a HTCondor que ejecute el trabajo con los ajustes anteriores.

Tabla 16: Trabajo 1 "submit.sub".

[41] **Universo**: En el contexto de HTCondor el universo se refiere al entorno de ejecución para un trabajo. Cada universo tiene sus propias características, restricciones y casos de uso específicos. La elección del universo depende del tipo de trabajo que se va a ejecutar y de los requisitos particulares de ejecución del trabajo. Aquí una breve explicación de los diferentes universos:
Vanilla Universe: Adecuado para programas simples y secuenciales, así como scripts de shell.
Grid Universe: Diseñado para ejecutar trabajos en sistemas de gestión remota, enviandolos a la grid.
Java Universe: Permite ejecutar programas de la Máquina Virtual de Java (JVM).
Scheduler Universe: Permite enviar trabajos ligeros, se ejecutan inmediatamente en la máquina desde la que se envía.
Local Universe: Permite la ejecución inmediata de trabajos en la máquina desde la que se envían, sin esperar asignación a una máquina remota, evaluando los requisitos contra el condor_schedd.
Parallel Universe: Diseñado para programas paralelos como trabajos MPI, permitiendo la ejecución en múltiples máquinas para un solo trabajo.
VM Universe: Facilita la ejecución de máquinas virtuales KVM y Xen dentro del entorno de HTCondor.

Esta primera prueba es similar a “Ejecución de un proceso en el nodo HTCondor-CE (submit) indicado: *condor_ce_run*.” del apartado anterior con la diferencia de que en este ejercicio se presenta el archivo “submit.sub” el cual describe el trabajo como se muestra en la tabla 16. Se muestran a continuación los comandos y el resultado del trabajo , ver Imagen 69.

prueba4@submit$ condor_ce_submit -remote submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 hostname.sub // Se envia el trabajo, usando el archivo descriptivo hostname.sub
prueba4@submit$ condor_ce_q 23 // Muestra el estado del trabajo

prueba4@submit$ condor_transfer_data -name submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 23 // Transfiere los archivos resultado que se describen en hostname.sub

prueba4@submit$ cat hostname.out // Muestra el resultado del trabajo, en este caso el nodo que respondió al comando /bin/hostname.

```
[prueba4@submit semana_htcondor]$ condor_ce_submit -remote submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 hostname.sub
Submitting job(s).
1 job(s) submitted to cluster 23.
[prueba4@submit semana_htcondor]$ condor_ce_q 23


-- Schedd: submit.atmosfera.unam.mx : <132.248.8.159:9619?... @ 02/12/24 13:55:19
OWNER         BATCH_NAME     SUBMITTED   DONE   RUN    IDLE  TOTAL JOB_IDS
usuarioNormal ID: 23       2/12 13:54      _      _      1      1 23.0

Total for query: 1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
Total for all users: 10 jobs; 2 completed, 2 removed, 2 idle, 0 running, 4 held, 0 suspended

[prueba4@submit semana_htcondor]$ condor_transfer_data -name submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 23
Fetching data files...
[prueba4@submit semana_htcondor]$ ls
hostname  hostname.err  hostname.out  hostname.sub
[prueba4@submit semana_htcondor]$ cat hostname.out
nodo1
[prueba4@submit semana_htcondor]$ condor_ce_ping -name submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 -type schedd -table write read
         Instruction Authentication Encryption Integrity Decision Identity
               write      SCITOKENS        AES       AES    ALLOW usuarioNormal@users.htcondor.org
                read      SCITOKENS        AES       AES    ALLOW usuarioNormal@users.htcondor.org
[prueba4@submit semana_htcondor]$ cat hostname.sub
executable = /bin/hostname

output = hostname.out
error = hostname.err
log = hostname.log

request_cpus = 1
request_memory = 1MB
request_disk = 1MB

queue
[prueba4@submit semana_htcondor]$
```

Imagen 69: Prueba de Envío del trabajo 1: “submit.sub”

5.2.2. Prueba de Envío del trabajo 2: “Test-script.sh”

Creamos el programa “test-script.sh” y le damos permisos de ejecución. test-script es un script que contiene la información que se va a realizar y al cual se hace referencia con el archivo test-script.sub el cual contiene la información respecto a los requisitos para ejecutar el trabajo, ver tabla 17. [51]

Trabajo 2 “Test-script.sh”	
Programa “test-script.sh”	Descripción
#!/bin/sh # START echo 'Date: ' `date` echo 'Host: ' `hostname` echo 'System: ' `uname -spo` # Imprime el nombre del programa/script que se está ejecutando echo "Program: $0" #Imprime los argumentos pasados al script. echo "Args: $*" # Lista los archivos en el directorio actual echo 'ls: ' `ls` # END	Este script imprime información como la fecha y hora, el nombre del host, información del sistema operativo, el nombre del script y los argumentos pasados al script, así como la lista de archivos en el directorio actual.
Programa “test-script.sub”	Descripción
executable = test-script.sh **arguments** = foo bar baz **output** = script.out **error** = script.err **log** = script.log **request_cpus** = 1 **request_memory** = 1MB **request_disk** = 1MB queue	Este archivo hace referencia al script a ejecutar, muestra los argumentos que recibe el script, nombra los 3 diferentes archivos de salida (script.out, script.err y script.log) y los requerimientos para correr el programa.

Tabla 17: Programas del Trabajo 2: “Test-script.sh”.

De manera similar al ejemplo anterior mandamos el archivo, traemos los archivos de salida y visualizamos el archivo de salida script.out

A continuación mostramos el procedimiento completo desde el levantamiento del oidc-agente, como se carga la cuenta de oidc, se expide el token, como se guarda el token en la variable de entorno BEARER_TOKEN, mandar el trabajo, traer los archivos de salida, la visualización de la propia salida que genera el trabajo y verificar si se usa scitokens como método de autenticación.

Pasos previos al envío del trabajo HTCondor:
prueba4@submit $ eval $(oidc-agent) // Se inicia el agente
prueba4@submit $ oidc-add prueba01 // Se carga la cuenta oidc previamente configurada
prueba4@submit $ export BEARER_TOKEN=$(oidc-token --aud=submit.atmosfera.unam.mx:9619 prueba01) // Se asigna el valor del token a la variable de entorno BEARER_TOKEN

Envío del trabajo a la grid y consulta del estado por medio del ID job:
prueba4@submit $ condor_ce_submit -remote submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 test-script.sub // Se envia el trabajo y en consola vemos el ID job = 27
prueba4@submit $ condor_ce_q 27 // Se consulta el estado del trabajo por medio de su ID

Recuperación de los resultados, impresión de resultados en consola y prueba de autenticación:
prueba4@submit $ condor_transfer_data -name submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 27 // Transferimos los archivos resultantes del trabajo.
prueba4@submit $ cat script.out // Imprimimos en consola el resultado del trabajo
prueba4@submit $ condor_ce_ping -name submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 -type schedd -table write read // Se verifica que se usó el token como método de autenticación

Es crucial destacar que los comandos previamente mencionados ilustran el procedimiento completo (ver imagen 70) del levantamiento del oidc-agente, la carga de la cuenta oidc (previamente configurada) para obtener el token. El envío de un trabajo HTCondor y verificar los resultados del mismo. En los ejemplos subsiguientes, se omitirán los pasos preliminares para obtener el token, y se enfocará en el envío del trabajo y la recuperación de los archivos resultantes. Se asumirá que la configuración del agente OIDC, la carga de la cuenta OIDC y la obtención del token se han realizado previamente como parte del proceso de preparación para enviar un trabajo HTCondor.

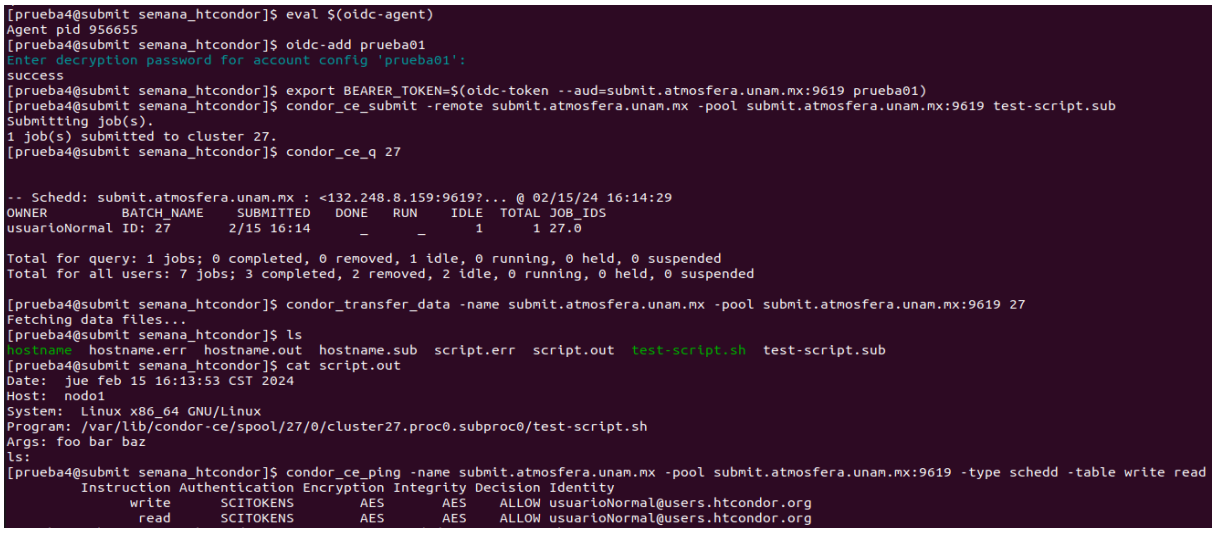

Imagen 70: Prueba de Envío del trabajo 2: “test-script.sh”.

5.2.3. Prueba de Envío del trabajo 3: “programa_hola_mundo.sh”

El siguiente trabajo representa el “hola mundo” para Grid. El “programa_hola_mundo.sh” en bash imprime información sobre el nodo que realiza el trabajo. De este trabajo se destaca el uso de la función dentro del programa bash. En el “archivo_descriptivo_envio.sh” se hace referencia a $(Cluster) y $(Process), ver tabla 18. Estas dos funciones son importantes en la realización de trabajos HTCondor. $(Cluster) nos da el ID del trabajo. $(Process) nos da la instancia del trabajo. Puede ser que deseemos ejecutar el mismo trabajo varias veces y para evitar una sobreescritura en los archivos de salida se usan estas variables que darán el nombre correspondiente a cada archivo de salida evitando la sobreescritura.

Trabajo 3 “programa_hola_mundo.sh”	
Programa “programa_hola_mundo.sh”	Descripción

#!/bin/bash
#Definicion de la funcion "_mensaje()"
_mensaje(){
 	
echo "$(date +%r) ::+++:: $*"
 	
}
_mensaje "Nombre del servidor que atiende el trabajo"
# Comandos a ejecutar 	
hostnamectl
sleep 1m
exit 0
	Este script imprime la hora en formato am/pm, información del sistema (hostname) incluido el nombre del servidor y el script se detiene un minuto.

Programa “archivo_descriptivo_envio.sh”	Descripción
universe = vanilla # Files **executable** = programa_hola_mundo.sh **output** = resultado.$(Cluster).$(Process).out **error** = resultado.$(Cluster).$(Process).err **log** = resultado.$(Cluster).$(Process).log **ShouldTransferFiles** = YES **WhenToTransferOutput** = ON_EXIT # Submit a single job queue	Este archivo determina el universo a utilizar, hace referencia al script a ejecutar “programa_hola_mundo.sh”, nombra los 3 diferentes archivos de salida. **ShouldTransferFiles** = YES Indica si HTCondor debe transferir archivos entre el nodo de ejecución y el nodo de envío. **WhenToTransferOutput** = ON_EXIT Especifica cuándo se transferirá la salida del trabajo de vuelta al nodo de envío.

Tabla 18: Programas del Trabajo 3: “programa_hola_mundo.sh”.

Los comandos para enviar el trabajo y recuperar resultados htcondor son:

prueba4@submit $ condor_ce_submit -remote submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 archivo_descriptivo_envio.sh // Se envia el trabajo

prueba4@submit $ condor_transfer_data -name submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 17 // Se recuperan los resultados del trabajo 17

Podemos ver el resultado del trabajo, el cual es el archivo resultado.17.0.out donde 17 es el ID job del trabajo y el número cero representa la instancia del programa. Como podemos ver en la imagen 71 vemos los resultados lo cuales son la impresión de la hora en formato am/pm datos del nodo que atendió el trabajo.

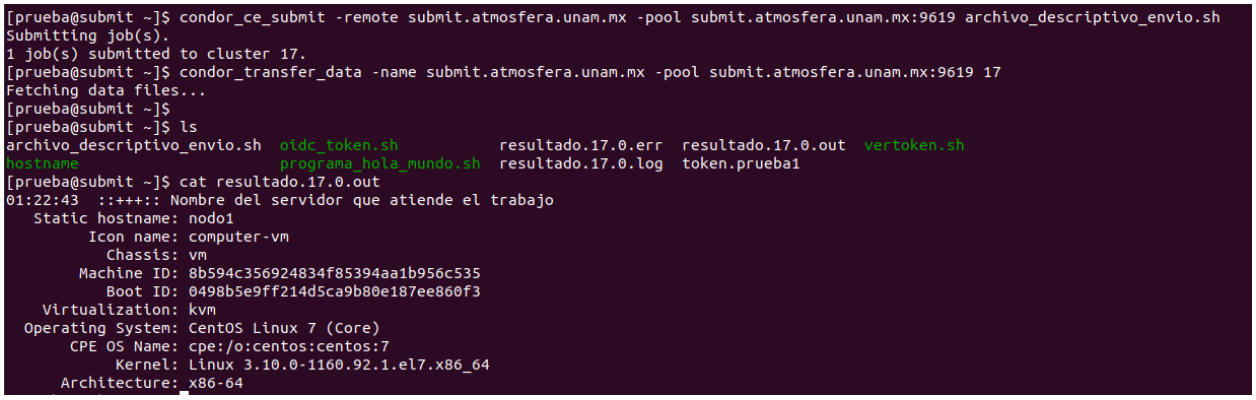

Imagen 71: Prueba de Envío del trabajo 3: “programa_hola_mundo.sh”.

Es posible visualizar la información por medio del comando condor_ce_q, el cual nos muestra el estado de los trabajos en cola. Este comando se puede usar con la opción -batch y -nobatch. Ambas opciones difieren en el cómo presentan la información. Por ejemplo, la opción batch agrupa la información de la cola de trabajos por lotes de trabajos. Si un trabajo tiene alguna instancia no se mostrará dicha instancia.En otras palabras no se repetirá el ID del trabajo. La opción -nobatch muestra información detallada de cada trabajo individual, sus instancias incluso el archivo o parámetros del trabajo. Esta opción permite ver el desglose de cada trabajo con sus respectivas instancias. ver imagen 72.

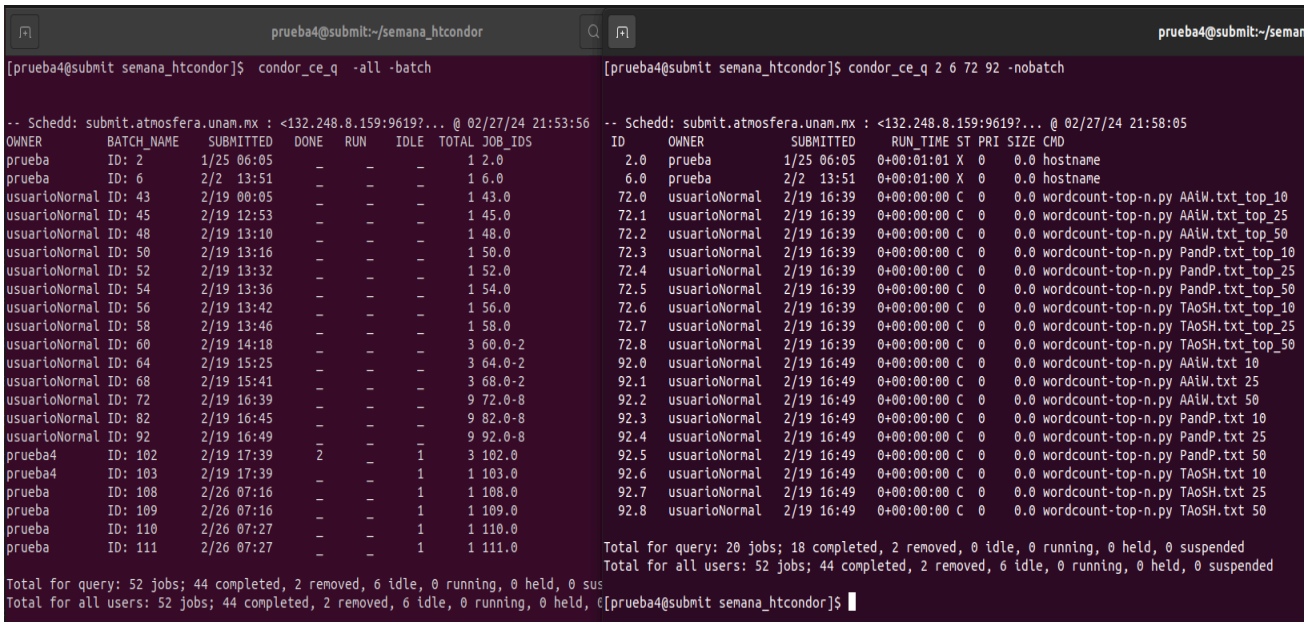

Imagen 72: condor_ce_q opciones: “-batch” y “-nobatch”.

5.2.4. Prueba de Envío del trabajo 4: ”ejercicio_simple_en_C.c”

A continuación mostramos un trabajo que consiste en enviar un programa simple escrito en C. Para enviar este trabajo es importante compilar previamente el programa “ejercicio_simple_en_C.c” ya que de no ser asi podria existir problemas de compatibilidad debido a que el nodo que ejecute el programa podría o no contar con las bibliotecas o versión de C que requiere el programa. Ahora contaremos tres archivos: el archivo de programa “*ejercicio_simple_en_C.c*”, el archivo compilado del programa “*ejercicio_En_C.out*”, el archivo descriptor “*ejercicio_simple_en_C.sub*”, ver tabla 19.

Trabajo 4 “ejercicio_simple_en_C.c”	
Programa “ejercicio_simple_en_C.c”	Descripción

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(int argc, char **argv)
{
int sleep_time;
int input;
int failure; //bandera de salida: 0 = exito, 1= error.
// Verifica si se proporcionan los argumentos necesarios
if (argc != 3) {
printf("Proporciona los siguientes argumentos: <sleep-time> <integer>\n");
failure = 1;
} else {
// Convierte los argumentos en enteros
sleep_time = atoi(argv[1]); // Convierte el primer argumento
input = atoi(argv[2]); // Convierte el segundo argumento
// Muestra un mensaje indicando que el programa está pensando
printf("Thinking really hard for %d seconds...\n", sleep_time);
// Hace que el programa duerma durante el tiempo especificado
sleep(sleep_time);
// Imprime el resultado del cálculo (el doble del segundo argumento)
printf("We calculated: %d\n", input * 2);
failure = 0; // Marca que no hubo fallos }
return failure; // Devuelve 0 = exitoso, 1 = fallo }
	Este programa espera dos argumentos de línea de comandos: un tiempo de espera en segundos y un número entero. Luego, el programa espera el tiempo especificado por el primer argumento y calcula el doble del segundo argumento el cual muestra en pantalla.

Programa “archivo_descriptivo_envio.sh”	Descripción
executable = ejercicio_En_C.out **arguments** = "60 64" **output** = programa_en_C.out **error** = programa_en_C.err **log** = programa_en_C.log **should_transfer_files** = YES **when_to_transfer_output** = ON_EXIT **request_cpus** = 1 **request_memory** = 7MB **request_disk** = 1MB queue	Este archivo descriptivo hace referencia al archivo compilado, el cual será ejecutado con los argumentos 60, 64. Especifica cómo se tratan los archivos y los requerimientos computacionales.

Tabla 19: Programas del Trabajo 4: “ejercicio_simple_en_C.c”.

Los comandos de envío y recuperación de archivos son:

prueba@submit_$ condor_ce_submit -remote submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 ejercicio_simple_en_C.sub // Se envia el trabajo

prueba@submit_$ condor_transfer_data -name submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 112 // Se recuperan los archivos

prueba@submit_$ cat c-program.out // ver los resultados

Para este ejercicio fue necesario previamente obtener el token, exportar la variable del token, compilar el programa, ejecutar el programa realizando la prueba para ver el resultado y compararlo al momento de enviar el trabajo y obtener los resultados. En este caso los argumentos son 60 para el tiempo de sleep y 64 para realizar el cálculo, el cual nos regresa 128, ver imagen 73.

```
[prueba@submit ~]$ gcc ejercicio_simple_en_C.c -o ejercicio_En_C.out
[prueba@submit ~]$ ./ejercicio_En_C.out 60 64
Thinking really hard for 60 seconds...
We calculated: 128
[prueba@submit ~]$ oidc-add prueba1
Error: Could not get the socket path from env var 'OIDC_SOCK'. Have you set the env var?

[prueba@submit ~]$ eval $(oidc-agent)
Agent pid 1886445
[prueba@submit ~]$ oidc-add prueba1
Enter decryption password for account config 'prueba1':
success
[prueba@submit ~]$ export BEARER_TOKEN=$(oidc-token --aud=submit.atmosfera.unam.mx:9619 prueba1)
[prueba@submit ~]$ gcc ejercicio_simple_en_C.c -o ejercicio_En_C.out
[prueba@submit ~]$ ./ejercicio_En_C.out 60 64
Thinking really hard for 60 seconds...
We calculated: 128
[prueba@submit ~]$ ls
archivo_descriptivo_envio.sh  c-program.err  ejemplo_py_avanzado  ejercicio_En_C.out       ejercicio_simple_en_C.sub  programa_hola_mund
compilarEnC.txt               c-program.log  ejemplos_avanzados   ejercicio_simple_en_C.c  oidc_token.sh              token.prueba1
[prueba@submit ~]$ condor_ce_submit -remote submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 ejercicio_simple_en_C.sub
Submitting job(s).
1 job(s) submitted to cluster 112.
[prueba@submit ~]$ condor_ce_q 112


-- Schedd: submit.atmosfera.unam.mx : <132.248.8.159:9619?... @ 02/27/24 22:26:35
OWNER         BATCH_NAME    SUBMITTED   DONE   RUN    IDLE  TOTAL JOB_IDS
usuarioNormal ID: 112      2/27 22:26      _      _      1      1 112.0

Total for query: 1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
Total for all users: 54 jobs; 44 completed, 2 removed, 8 idle, 0 running, 0 held, 0 suspended

[prueba@submit ~]$ condor_transfer_data -name submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 112
Fetching data files...
[prueba@submit ~]$ ls
archivo_descriptivo_envio.sh  c-program.err  ejemplo_py_avanzado  ejercicio_En_C.out       ejercicio_simple_en_C.sub  programa_en_C.err
compilarEnC.txt               c-program.log  ejemplos_avanzados   ejercicio_simple_en_C.c  oidc_token.sh              programa_en_C.out
[prueba@submit ~]$ cat programa_en_C.out
Thinking really hard for 60 seconds...
We calculated: 128
[prueba@submit ~]$
```

Imagen 73: Prueba de Envío del trabajo 4: “ejercicio_simple_en_C.c”.

5.2.5. Prueba de Envío del trabajo 5: “calculaPI_Instancias.c”

El programa “calculaPI_Instancias.c” es un programa que calcula por medio del método de Monte carlo el número PI. Es un método usado para aproximar expresiones matemáticas complejas y costosas de evaluar con exactitud. El nombre Monte Carlo proviene de la similitud de los métodos de simulación estadística con la aleatoriedad de los juegos de azar. Monte Carlo, la capital de Mónaco, es un centro de juegos de azar.

El método de montecarlo consiste en:

1. Generar puntos aleatorios dentro de un cuadro, ver imagen 74.
2. Contar los puntos que caen dentro de un círculo inscrito en el cuadro antes mencionado.
3. La relación entre el área del círculo y el área del cuadrado es proporcional a $\frac{\pi}{4}$.

$$\frac{Área\ del\ círculo}{Área\ del\ rectángulo} = \frac{\pi r^2}{L \times L} \quad \text{Si r=1 y L=2} \rightarrow \frac{Área\ del\ círculo}{Área\ del\ rectángulo} = \frac{\pi}{4}$$

4. Por lo cual, al multiplicar esta relación por 4, se obtiene una estimación de Pi.

$$\frac{4\times\ Área\ del\ círculo}{Área\ del\ rectángulo} = \pi \Rightarrow \frac{4\times(Número\ de\ puntos\ dentro\ del\ círculo)}{Número\ de\ puntos\ totales} = \pi$$

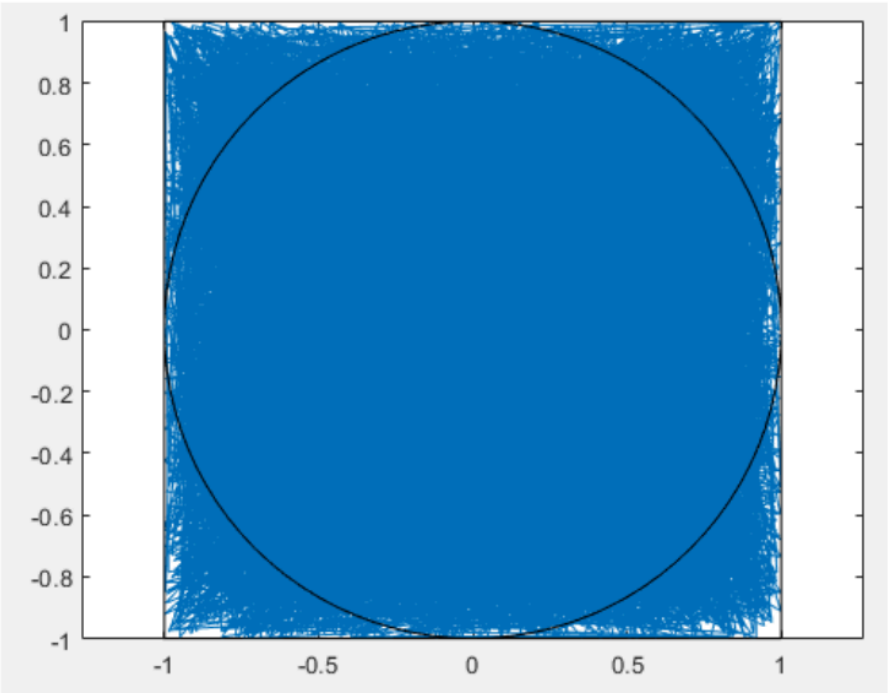

Imagen 74: Representación gráfica de los puntos aleatorios para el método de Monte Carlo.

Se muestra el programa que se utilizó así como el archivo descriptivo del trabajo HTCondor, ver tabla 20.

<table>
<tr><th colspan="2">Trabajo 5 “calculaPI_Instancias.c”</th></tr>
<tr><th>Programa “calculaPI_Instancias.c”</th><th>Descripción</th></tr>
<tr><td>

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
int main(int argc, char *argv[])
{ struct timeval mi_tiempo;
  int iteraciones = 0;
  int dentro_circulo = 0;
  int i;
  double x, y, estimacion_pi;
//Funciones:
 gettimeofday(&mi_tiempo, NULL);
 srand48(mi_tiempo.tv_sec ^ mi_tiempo.tv_usec);
  if (argc == 2) {
            iteraciones = atoi(argv[1]);
  } else {
            printf("uso: circlepi ITERACIONES\n");
            exit(1);
 }
  for (i = 0; i < iteraciones; i++) {
            x = (drand48() - 0.5) * 2.0;
            y = (drand48() - 0.5) * 2.0;
            if (((x * x) + (y * y)) <= 1.0) {
            dentro_circulo++;
            }
  }
estimacion_pi = 4.0 * ((double) dentro_circulo / (double) iteraciones);
printf("%d iteraciones, %d dentro; pi = %f\n", iteraciones, dentro_circulo,
estimacion_pi);
  return 0;
}
```

</td><td>El programa toma un argumento de línea de comandos, este número especifica el número de iteraciones.

La función gettimeofday genera una semilla de inicialización para la función de generación de números aleatorios.

Después se realiza un for usando el número de iteraciones como límite. En cada iteración se generan las coordenadas aleatorias X,Y. Si las coordenadas están dentro del círculo se incrementa el contador “inside_circle”.

El cálculo de PI se realiza con:

$PI=4 \frac{\text{(Número de puntos dentro del círculo)}}{\text{Número de puntos totales}}$</td></tr>
<tr><th>Programa “circuloPI_descriptor.sub”</th><th>Descripción</th></tr>
<tr><td>

executable = calculaPI_compilado.out
arguments = 1000000

output = calculaPI_$(Cluster)_$(Process).out
error = calculaPI_$(Cluster)_$(Process).err
log = calculaPI_registro.log

request_cpus = 2
request_memory = 10MB
request_disk = 10MB

queue 10

</td><td>Este archivo descriptivo ejecuta el archivo compilado “calculaPI_compilado.out”, como argumento tenemos un millón de iteraciones. Se reserva los recursos computacionalmente necesarios.
Los archivos de salida mostrarán el ID job y las instancias del programa. Al final se realizarán 10 instancias del programa. En otras palabras, correremos 10 veces el mismo programa (queue 10).</td></tr>
</table>

Tabla 20: Programas del Trabajo 5: “calculaPI_Instancias.c”.

El procedimiento es igual al ejercicio anterior, se debe de obtener el compilado del programa en C. Al cual se hace referencia en el descriptor del trabajo. Una vez obtenido el compilado podemos enviar el trabajo HTCondor y verificar los diez resultados generados de correr el programa diez veces. Esto abre la puerta al dilema entre la elección entre enviar un solo trabajo con un gran número de iteraciones o enviar múltiples trabajos con un menor número de iteraciones cada uno. Es algo importante a considerar para optimizar los trabajos que se envían a la grid.

En la imagen 75 mostramos el resultado de enviar el trabajo 5. En la imagen podemos:

- ☑ La generación y exportación del token a la variable BEARER_TOKEN.
- ☑ Obtención del archivo objeto "calculaPI_compilado.out".
- ☑ Una prueba del funcionamiento del programa.
- ☑ Envío del trabajo.
- ☑ Recuperación de los resultados.
- ☑ Mostrar los resultados en consola.

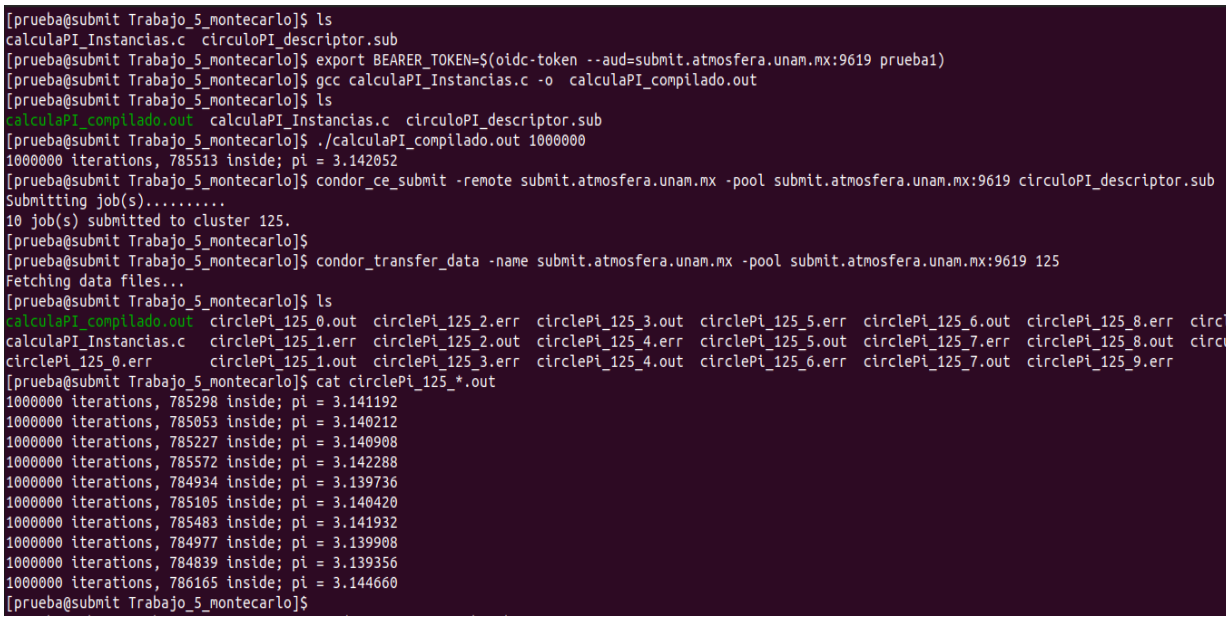

Imagen 75: Prueba de Envío del trabajo 5: "calculaPI_Instancias.c".

Nuestro trabajo, identificado con el ID 125, se ejecutaron 10 procesos según lo especifica el archivo "circuloPI_descriptor.sub" utilizando la opción "*queue 10*".
Es en este punto donde se demuestra la utilidad de las variables $(cluster) y $(Process) dentro del archivo "circuloPI_descriptor.sub":

output = circlePi_$(Cluster)_$(Process).out
error = circlePi_$(Cluster)_$(Process).err

Sin el uso de estas variables, solo tendríamos un único archivo de salida, debido a la sobreescritura de resultados. El uso de $(Cluster) y $(Process) nos permite generar archivos de salida únicos para cada trabajo, facilitando la gestión y análisis de los resultados obtenidos.

Este ejercicio nos permite ver el comportamiento de otro comando que es condor_ce_q con la opción -nobatch.

prueba@submit $ condor_ce_q 125 -nobatch // ver los resultados sin acomodar por lotes.

Este comando muestra la información[42] detallada del trabajo 125 sin agrupar por lotes, ver imagen 76. Nos muestra los diez procesos "instancias" del trabajo 125 así como el argumento utilizado por cada uno de los procesos, que en nuestro caso es 1 millón.

```
[prueba@submit Trabajo_5_montecarlo]$ condor_ce_q 125 -nobatch


-- Schedd: submit.atmosfera.unam.mx : <132.248.8.159:9619?... @ 02/29/24 02:42:56
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
 125.0   usuarioNormal   2/29 02:41   0+00:00:00 C  0    0.0 calculaPI_compilado.out 1000000
 125.1   usuarioNormal   2/29 02:41   0+00:00:00 C  0    0.0 calculaPI_compilado.out 1000000
 125.2   usuarioNormal   2/29 02:41   0+00:00:00 C  0    0.0 calculaPI_compilado.out 1000000
 125.3   usuarioNormal   2/29 02:41   0+00:00:00 C  0    0.0 calculaPI_compilado.out 1000000
 125.4   usuarioNormal   2/29 02:41   0+00:00:00 C  0    0.0 calculaPI_compilado.out 1000000
 125.5   usuarioNormal   2/29 02:41   0+00:00:00 C  0    0.0 calculaPI_compilado.out 1000000
 125.6   usuarioNormal   2/29 02:41   0+00:00:00 C  0    0.0 calculaPI_compilado.out 1000000
 125.7   usuarioNormal   2/29 02:41   0+00:00:00 C  0    0.0 calculaPI_compilado.out 1000000
 125.8   usuarioNormal   2/29 02:41   0+00:00:00 C  0    0.0 calculaPI_compilado.out 1000000
 125.9   usuarioNormal   2/29 02:41   0+00:00:00 C  0    0.0 calculaPI_compilado.out 1000000

Total for query: 10 jobs; 10 completed, 0 removed, 0 idle, 0 running, 0 held, 0 suspended
Total for all users: 82 jobs; 74 completed, 2 removed, 6 idle, 0 running, 0 held, 0 suspended
```

Imagen 76: Comando condor_ce_q con la opción -nobatch.

[42] La información del comando condor_ce_q -nobatch:
ID: Identificación del trabajo.OWNER: usuario del propietario que envió el trabajo.
SUBMITTED: Fecha y hora de envío del trabajo.
RUN_TIME: Tiempo total transcurrido desde que el trabajo comenzó a ejecutarse, en (días + horas:minutos: segundos)
ST: Estado del trabajo:
- **I** "idle" (esperando ejecutarse)
- **R** "Running" (en ejecución)
- **H** "held" (suspendido)

PRI: Prioridad del trabajo.
SIZE: Uso actual de memoria en tiempo de ejecución, en megabytes.
CMD: Comando ejecutable (con argumentos).

5.2.6. Prueba de Envío del trabajo 6: ”wordcount-top-n.py”

Este ejercicio basado en el taller “OSG School 2023” [53], considerado el más completo involucra:

- ☑ 3 libros en formato de texto: 'AAiW.txt', 'PandP.txt' y 'TAoSH.txt'
- ☑ 1 archivo “books_n.txt”.
- ☑ 1 programa “wordcount-top-n.py”
- ☑ 1 archivo descriptor “wordcount-top-n.sub”

El archivo “*books_n.txt*” contiene una lista de argumentos que será utilizada por el programa “*wordcount-top-n.py*”.

El programa “*wordcount-top-n.py*” es un script en Python2 que cuenta la frecuencia de las palabras en un archivo de texto y luego muestra las palabras más comunes junto con su frecuencia. Para su funcionamiento requiere de dos argumentos: el archivo de texto y un número. El número refleja el top de las palabras más frecuentes encontradas en el libro, ver tabla 21.

El archivo descriptor “wordcount-top-n.sub “ describe los requerimientos computacionales necesarios así como los argumentos que requiere el programa para ser enviado a la grid, ver tabla 21.

Trabajo 6 “*wordcount-top-n.py*”	
Programa “*wordcount-top-n.py*”	Programa “*wordcount-top-n.sub*”

#!/usr/bin/env python
import os
import sys
import operator
 	
if len(sys.argv) != 3:
print 'Usage: {} DATA NUM_WORDS'.format(os.path.basename(sys.argv[0]))
sys.exit(1)
input_filename = sys.argv[1]
num_words = int(sys.argv[2])
 	
words = {}
 	
with open(input_filename, 'r') as my_file:
for line in my_file:
line_words = line.split()
for word in line_words:
if word in words:
words[word] += 1
else:
words[word] = 1
 	
sorted_words = sorted(words.items(), key=operator.itemgetter(1))
for word in sorted_words[-num_words:]:
print '{} {:8d}'.format(word[0], word[1])
	executable = wordcount-top-n.py
	arguments = "$(book) $(n)"
	
	output = $(book)_top_$(n).out
	error = $(book)_$(n).err
	log = wordcount.log
	
	transfer_input_files = $(book)
	
	request_cpus = 2
	request_memory = 100MB
	request_disk = 25MB
	
	queue **book, n** from **books_n.txt**

Tabla 21: Programas del Trabajo 6: “wordcount-top-n.py”.

Este ejercicio fue parte de un curso “Escuela OSG 2023” [51] impartido entre el 7 y 11 agosto del 2023. En este curso se desarrollan diversos ejercicios sobre la utilización de HTCondor.
Obtenemos los libros a partir de:
prueba@submit $ wget http://proxy.chtc.wisc.edu/SQUID/osgschool20/books.zip
prueba@submit $ unzip books.zip // Se descomprimen los archivos
Como podemos observar los libros contienen una gran cantidad de líneas, palabras que nuestro programa deberá procesar, ver imagen 77.

```
[prueba@submit ejemplo_py_avanzado]$ ls -lh
total 8.0K
-rwxr-xr-x 1 prueba prueba 645 feb 19 16:49 wordcount-top-n.py
-rw-rw-r-- 1 prueba prueba 260 feb 19 16:48 wordcount-top-n.sub
[prueba@submit ejemplo_py_avanzado]$ wget http://proxy.chtc.wisc.edu/SQUID/osgschool20/books.zip
--2024-02-29 05:25:17--  http://proxy.chtc.wisc.edu/SQUID/osgschool20/books.zip
Resolviendo proxy.chtc.wisc.edu (proxy.chtc.wisc.edu)... 128.104.100.219, 128.104.100.171, 128.104.100.172,
Conectando con proxy.chtc.wisc.edu (proxy.chtc.wisc.edu)[128.104.100.219]:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 540785 (528K) [application/zip]
Grabando a: “books.zip”

100%[=============================================================================================================

2024-02-29 05:25:17 (1.61 MB/s) - “books.zip” guardado [540785/540785]

[prueba@submit ejemplo_py_avanzado]$ ls -lh
total 540K
-rw-rw-r-- 1 prueba prueba 529K ago  6  2023 books.zip
-rwxr-xr-x 1 prueba prueba  645 feb 19 16:49 wordcount-top-n.py
-rw-rw-r-- 1 prueba prueba  260 feb 19 16:48 wordcount-top-n.sub
[prueba@submit ejemplo_py_avanzado]$ unzip books.zip
Archive:  books.zip
  inflating: AAiW.txt
  inflating: PandP.txt
  inflating: TAoSH.txt
[prueba@submit ejemplo_py_avanzado]$ ls -lh
total 2.0M
-rw-rw-r-- 1 prueba prueba 160K ago  6  2023 AAiW.txt
-rw-rw-r-- 1 prueba prueba 529K ago  6  2023 books.zip
-rw-rw-r-- 1 prueba prueba 688K ago  6  2023 PandP.txt
-rw-rw-r-- 1 prueba prueba 569K ago  6  2023 TAoSH.txt
-rwxr-xr-x 1 prueba prueba  645 feb 19 16:49 wordcount-top-n.py
-rw-rw-r-- 1 prueba prueba  260 feb 19 16:48 wordcount-top-n.sub
[prueba@submit ejemplo_py_avanzado]$
[prueba@submit ejemplo_py_avanzado]$ wc AAiW.txt
  3735  29461 163780 AAiW.txt
[prueba@submit ejemplo_py_avanzado]$ wc -l AAiW.txt
3735 AAiW.txt
[prueba@submit ejemplo_py_avanzado]$ wc -l AAiW.txt PandP.txt TAoSH.txt
   3735 AAiW.txt
  13426 PandP.txt
  13053 TAoSH.txt
  30214 total
```

Imagen 77: Obtención de los archivos para el trabajo 6.

Para enviar el trabajo usamos:
prueba@submit $ condor_ce_submit -remote submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 wordcount-top-n.sub // enviando el trabajo htcondor-ce

Ahora consultamos el trabajo enviado y mostramos los procesos de nuestro trabajo enviado, ver imagen 78:
prueba@submit $ condor_ce_q 136 -nobatch // consultar el trabajo 136 y sus procesos “instancias”

```
[prueba@submit ejemplo_py_avanzado]$ ls -l
total 1964
-rw-rw-r-- 1 prueba prueba 163780 ago  6  2023 AAiW.txt
-rw-rw-r-- 1 prueba prueba    128 feb 29 13:06 books_n.txt
-rw-rw-r-- 1 prueba prueba 540785 ago  6  2023 books.zip
-rw-rw-r-- 1 prueba prueba 704145 ago  6  2023 PandP.txt
-rw-rw-r-- 1 prueba prueba 581888 ago  6  2023 TAoSH.txt
-rwxr-xr-x 1 prueba prueba    645 feb 19 16:49 wordcount-top-n.py
-rw-rw-r-- 1 prueba prueba    260 feb 19 16:48 wordcount-top-n.sub
[prueba@submit ejemplo_py_avanzado]$
[prueba@submit ejemplo_py_avanzado]$ export BEARER_TOKEN=$(oidc-token --aud=submit.atmosfera.unam.mx:9619 prueba1)
[prueba@submit ejemplo_py_avanzado]$ condor_ce_submit -remote submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 wordcount-top-n.sub
Submitting job(s).........
9 job(s) submitted to cluster 136.
[prueba@submit ejemplo_py_avanzado]$ condor_ce_q 136 -nobatch


-- Schedd: submit.atmosfera.unam.mx : <132.248.8.159:9619?... @ 02/29/24 13:07:43
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
 136.0   usuarioNormal   2/29 13:07   0+00:00:00 I  0    0.0 wordcount-top-n.py AAiW.txt 10
 136.1   usuarioNormal   2/29 13:07   0+00:00:00 I  0    0.0 wordcount-top-n.py AAiW.txt 25
 136.2   usuarioNormal   2/29 13:07   0+00:00:00 I  0    0.0 wordcount-top-n.py AAiW.txt 50
 136.3   usuarioNormal   2/29 13:07   0+00:00:00 I  0    0.0 wordcount-top-n.py PandP.txt 10
 136.4   usuarioNormal   2/29 13:07   0+00:00:00 I  0    0.0 wordcount-top-n.py PandP.txt 25
 136.5   usuarioNormal   2/29 13:07   0+00:00:00 I  0    0.0 wordcount-top-n.py PandP.txt 50
 136.6   usuarioNormal   2/29 13:07   0+00:00:00 I  0    0.0 wordcount-top-n.py TAoSH.txt 10
 136.7   usuarioNormal   2/29 13:07   0+00:00:00 I  0    0.0 wordcount-top-n.py TAoSH.txt 25
 136.8   usuarioNormal   2/29 13:07   0+00:00:00 I  0    0.0 wordcount-top-n.py TAoSH.txt 50

Total for query: 9 jobs; 0 completed, 0 removed, 9 idle, 0 running, 0 held, 0 suspended
Total for all users: 89 jobs; 63 completed, 2 removed, 24 idle, 0 running, 0 held, 0 suspended
```

Imagen 78: 1er parte, Prueba de Envío del trabajo 6.

En la imagen podemos ver las nueve ejecuciones de nuestro programa usando diferentes argumentos. Recordemos que nuestro programa necesita dos argumentos: El archivo de texto y el número que representa el top de palabras que más se repiten.

El archivo books_n.txt contiene la lista de argumentos que serán usados por el programa “wordcount-top-n.py”.

```
AAiW.txt, 10
AAiW.txt, 25
AAiW.txt, 50
PandP.txt, 10
PandP.txt, 25
PandP.txt, 50
TAoSH.txt, 10
TAoSH.txt, 25
TAoSH.txt, 50
```

Mediante el comando condor_ce_q 136 -nobatch podemos ver que nuestro programa fue ejecutado nueve veces con diferentes argumentos los cuales están en el archivo books_n.txt.

Esto sucedió debido a que en el “wordcount-top-n.sub” describimos el tipo de argumentos que recibiría nuestro programa en python: *arguments = "$(book) $(n)"* y estos argumentos serán buscados en nuestra lista “books_n.txt” desde la línea: *queue **book, n** from **books_n.txt.*** Es aquí donde se especifica el de donde tomará los argumentos.

Nuestro programa utilizará para la primera ejecución el archivo AAiW.txt y mostrará las 10 palabras que más se repiten en el archivo de texto. En la cuarta ejecución tomará el archivo de texto PandP.txt y mostrará las 10 palabras que más se repiten en el archivo de texto.

Ahora procedemos a recuperar los archivos y listarlos en consola, ver imagen 79:

prueba@submit $ condor_transfer_data -name submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 136

```
[prueba@submit ejemplo_py_avanzado]$ condor_transfer_data -name submit.atmosfera.unam.mx -pool submit.atmosfera.unam.mx:9619 136
Fetching data files...
[prueba@submit ejemplo_py_avanzado]$ ls -lh
total 2.0M
-rw-rw-r-- 1 prueba prueba 160K ago  6  2023 AAiW.txt
-rw-r--r-- 1 prueba prueba    0 feb 29 13:07 AAiW.txt_10.err
-rw-r--r-- 1 prueba prueba    0 feb 29 13:07 AAiW.txt_25.err
-rw-r--r-- 1 prueba prueba    0 feb 29 13:07 AAiW.txt_50.err
-rw-r--r-- 1 prueba prueba  125 feb 29 13:07 AAiW.txt_top_10.out
-rw-r--r-- 1 prueba prueba  318 feb 29 13:07 AAiW.txt_top_25.out
-rw-r--r-- 1 prueba prueba  656 feb 29 13:07 AAiW.txt_top_50.out
-rw-rw-r-- 1 prueba prueba  128 feb 29 13:06 books_n.txt
-rw-rw-r-- 1 prueba prueba 529K ago  6  2023 books.zip
-rw-rw-r-- 1 prueba prueba 688K ago  6  2023 PandP.txt
-rw-r--r-- 1 prueba prueba    0 feb 29 13:07 PandP.txt_10.err
-rw-r--r-- 1 prueba prueba    0 feb 29 13:07 PandP.txt_25.err
-rw-r--r-- 1 prueba prueba    0 feb 29 13:07 PandP.txt_50.err
-rw-r--r-- 1 prueba prueba  124 feb 29 13:07 PandP.txt_top_10.out
-rw-r--r-- 1 prueba prueba  315 feb 29 13:07 PandP.txt_top_25.out
-rw-r--r-- 1 prueba prueba  657 feb 29 13:07 PandP.txt_top_50.out
-rw-rw-r-- 1 prueba prueba 569K ago  6  2023 TAoSH.txt
-rw-r--r-- 1 prueba prueba    0 feb 29 13:07 TAoSH.txt_10.err
-rw-r--r-- 1 prueba prueba    0 feb 29 13:07 TAoSH.txt_25.err
-rw-r--r-- 1 prueba prueba    0 feb 29 13:07 TAoSH.txt_50.err
-rw-r--r-- 1 prueba prueba  124 feb 29 13:07 TAoSH.txt_top_10.out
-rw-r--r-- 1 prueba prueba  317 feb 29 13:07 TAoSH.txt_top_25.out
-rw-r--r-- 1 prueba prueba  644 feb 29 13:07 TAoSH.txt_top_50.out
-rwxr-xr-x 1 prueba prueba  645 feb 29 13:07 wordcount-top-n.py
-rw-rw-r-- 1 prueba prueba  260 feb 19 16:48 wordcount-top-n.sub
```

Imagen 79: 2da parte, Recuperación de archivos del trabajo 6.

Nos encontramos con un total de nueve archivos de salida. Distribuidos en tres conjuntos correspondientes a cada uno de los archivos de texto: “AAiW.txt”, “PandP.txt” y “TAoSH.txt”.[53]

Para cada archivo de texto se generaron tres archivos de salida correspondientes a los argumentos dados por el archivo “books_n.txt” Como no existieron errores todos los archivos con extensión “.err” están vacíos.
Para mostrar los resultados podremos usar el siguiente comando:

prueba@submit $ tail -n 3 AAiW.txt_top_*.out PandP.txt_top_*.out TAoSH.txt_top_*.out

De esta forma podremos visualizar las primeras tres líneas de los nueve de salida. El hecho de visualizar las tres palabras que más se repiten nos generará ver la misma salida para cada uno de los archivos de texto.

Por ejemplo, aunque el archivo de salida AAiw.txt_top_10.out contiene 10 líneas ordenadas de las palabras más frecuentes nosotros visualizamos las 3 palabras más frecuentes encontradas en el archivo de texto AAiW.txt. De la misma manera aunque el archivo de salida AAiw.txt_top_25.out contiene 25 líneas ordenadas de las palabras más frecuentes nosotros visualizamos las 3 palabras más frecuentes encontradas en el archivo de texto AAiW.txt. Por lo tanto nos debe mostrar el mismo resultado. ver imagen 80.

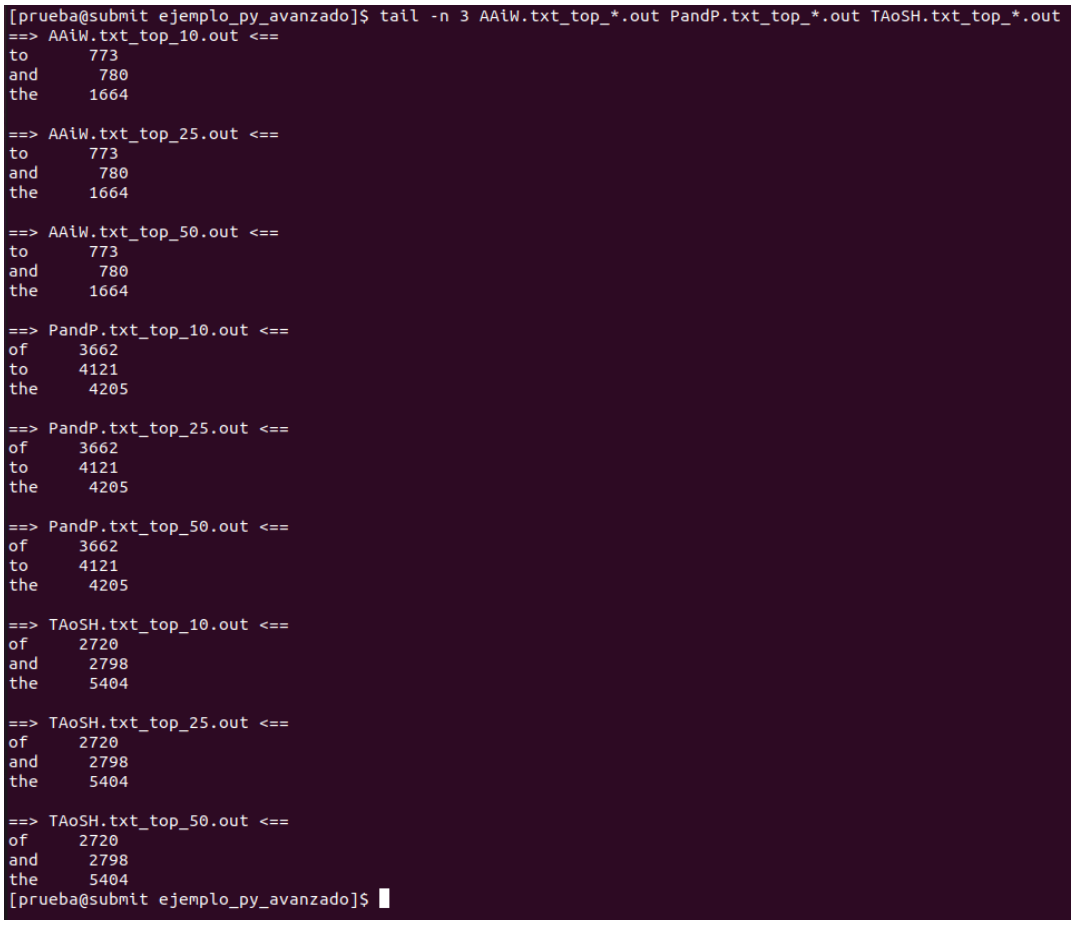

Imagen 80: 3era parte, Impresión de resultados del trabajo 6.

La forma de proceder normalmente nos indicaría que para cada ejecución del trabajo es necesario un archivo descriptor htcondor. Lo cual nos representará hacer nueve archivos descriptores. Para poder mandar cada uno de los nueve trabajos con sus respectivos argumentos. También implica mandar nueve trabajos, recuperar nueve veces los archivos. Justo en este ejemplo es donde se puede vislumbrar todo el trabajo en conjunto. Con este ejercicio validamos toda la instalación, configuración de la maqueta y de cada uno de sus componentes.

6. Conclusión

El presente proyecto se enfocó en la implementación de un sistema para la gestión, ejecución de trabajos HTCondor en un cluster virtual con Slurm, utilizando scitokens como método de autenticación para usuarios registrados en la Grid.

Se proporcionó un panorama general sobre Grids y redes académicas. Lo cual es fundamental para comprender el cómputo de alto rendimiento con un enfoque HTC.

La explicación de los diferentes conceptos como: x.509, JWT, HTCondor, SLURM, OAUTH, OIDC, INDIGO, NGINX. Estos conceptos, estándares resultan importantes ya que se usan en diferentes campos de la tecnología. La comprensión de estos conceptos, fue fundamental para brindar el contexto y las bases para la realización del proyecto.

La elaboración de esta maqueta supone la aplicación práctica de todos estos conceptos, lo que permite cumplir satisfactoriamente con los objetivos planteados al inicio del proyecto:

- ☑ **Objetivo principal:** Desarrollar un método para la autorización de usuarios a enviar trabajos por medio de tokens en un entorno Grid.
- ☑ **Objetivos secundarios:**
  - ☑ Investigar, examinar los conceptos: Redes Académicas, Tecnología Grid, HTC, HPC, JWT, X509, OAuth, INDIGO, OIDC.
  - ☑ Establecer una infraestructura virtual de un clúster.
  - ☑ Desarrollar una maqueta con la capacidad de utilizar tokens, enviar trabajos a un cluster y administrar usuarios.

La maqueta sigue la Configuración HTCondor-CE + Non-HTCondor Batch System. Es capaz de procesar trabajos slurm ya que es nuestro sistema de gestión de recursos para clusters. Es posible enviar trabajos por medio de htcondor-ce los cuales serán procesador por slurm.
Se realiza la autenticación de usuarios por medio de scitokens para el envío de trabajos htcondor. Los tokens permiten a los usuarios acceder a recursos específicos sin necesidad de proporcionar sus credenciales completas cada vez, esto mejora la seguridad. Los tokens tienen un tiempo de vida corto, son específicos del usuario lo que reduce el riesgo de uso indebido.

Por otra parte, los tokens son ampliamente utilizados en la actualidad como parte de estándares modernos de autenticación y autorización como OAUTH y OpenID Connect. La gran mayoría de los lenguajes de programación más utilizados brindan librerías para su implementación. Aunque el cambio de certificados x.509 a tokens fue obligatorio para el software htcondor. El uso de tokens permite seguir las mejores prácticas y estándares facilitando la interoperabilidad con otros sistemas y servicios.

7. Trabajo futuro.

Respecto a los trabajos a futuro podemos listar los siguientes:

1. Keycloak: Es otra opción a indigo IAM. Permite el uso de SSO, Inicio de sesión a través de redes sociales.Permite autenticar a los usuarios con los proveedores de identidad de OpenID Connect o SAML 2.0.
2. Compatibilidad OIDC versión 5:Durante la realización de la maqueta se trabajó con la versión 4 debido a que Grid UNAM ya trabajaba con esa versión y al utilizar la versión 5 existía un problema de compatibilidad. La nueva versión brinda mejoras de rendimiento, seguridad contra ataques Cross-Site Request Forgery, actualización de dependencias, soluciona un problema con la ruta del directorio tmp que podía ocurrir en algunos sistemas Windows y que impedía que se iniciara oidc-agent.
3. Mejorar la integración de los tokens para facilitar su uso: Aunque la obtención de tokens es una tarea sencilla por medio de comandos. La gran mayoría de usuarios finales no están familiarizados con ellos, lo cual podría complicar el uso de estos.
4. Autorización de recursos de almacenamiento/cómputo por medio de tokens: Es posible, la utilización de tokens para autorizaciones de recursos de cómputo, almacenamiento. Ya que hasta ahora se usan solo para autorizar a los usuarios el envío de trabajos a la grid.
5. Eficiencia al mandar trabajos: Uno de los conceptos, es el concepto de granularidad. El cual se refiere a la relación entre tiempo de comunicación y procesamiento. Este es un parámetro importante a conocer para mejorar la eficiencia entre la carga de procesamiento de cada trabajo enviado a la grid contra la cantidad de trabajos que se envían.
6. Incrementar la implementación del proyecto Grid: Como se mencionó este tipo de proyectos benefician a toda la comunidad. El hecho de incrementarla podría brindar el apoyo a aquellos institutos que no cuenten con el poder de cómputo necesario para sus investigaciones.
7. Colaboraciones: Una vez que se tiene mayor conocimiento, experiencia se podría colaborar de manera internacional/nacional en proyectos más grandes como se mencionó en el capítulo 1.

8. Anexos

Se muestran los archivos de configuración, programas y comandos utilizados.

8.1. Archivo de configuración slurm

Este archivo de configuración se encuentra en: /etc/slurm/slurm.conf y es el mismo para los nodos esclavos, nodo maestro y nodo submit.

```
ClusterName=cluster
ControlMachine=maestro
ControlAddr=172.17.100.100
AuthType=auth/munge
CryptoType=crypto/munge
ProctrackType=proctrack/cgroup
ReturnToService=1
SlurmctldPidFile=/var/run/slurm/slurmctld.pid
SlurmctldPort=6817
SlurmdPidFile=/var/run/slurm/slurmd.pid
SlurmdPort=6818
SlurmdSpoolDir=/var/spool/slurm/d
SlurmUser=root
StateSaveLocation=/var/spool/slurm/ctld
SwitchType=switch/none
TaskPlugin=task/none
InactiveLimit=0
KillWait=30
MinJobAge=300
SlurmctldTimeout=120
SlurmdTimeout=300
Waittime=0
SchedulerType=sched/backfill
SelectType=select/linear
##Debug
AccountingStorageType=accounting_storage/none
AccountingStoreJobComment=YES
JobCompType=jobcomp/filetxt                    #tipo de archivo: txt
JobCompLoc=/var/log/slurm/job_completions
JobAcctGatherFrequency=30
JobAcctGatherType=jobacct_gather/linux      #tipo de s.o
SlurmctldDebug=3
SlurmdDebug=3
#del archivo pdf instalación rápida de slurm con rpm
SlurmctldLogFile=/var/log/slurm/slurmctld.log
SlurmdLogFile=/var/log/slurm/slurmd.log
SlurmSchedLogFile=/var/log/slurm/slurmsched.log
SlurmSchedLogLevel=3
MpiDefault=none
#nodos 1 2 3
NodeName=nodo1 CPUs=4 Boards=1 SocketsPerBoard=1 CoresPerSocket=4 ThreadsPerCore=1 RealMemory=3788
NodeName=nodo2 CPUs=4 Boards=1 SocketsPerBoard=1 CoresPerSocket=4 ThreadsPerCore=1 RealMemory=3788
NodeName=nodo3 CPUs=4 Boards=1 SocketsPerBoard=1 CoresPerSocket=4 ThreadsPerCore=1 RealMemory=3788
#PARTICIONES
PartitionName=debug Nodes=ALL Default=YES MaxTime=INFINITE State=UP
```

8.2. Archivo de configuración Nginx inverso

```
       # Nginx configuration
server {
# Indica la ubicación del directorio raíz del sitio web
    root /var/www/iam.atmosfera.unam.mx;
# Enumera los archivos de índice por defecto que se utilizarían si se accede a una
# ruta sin especificar un archivo específico
    index index.html index.htm index.nginx-debian.html;
# Define los nombres de host que este servidor virtual debe manejar.
    server_name iam.atmosfera.unam.mx iam;
# Lugar donde se encuentran los registros "logs"
    access_log /var/log/nginx/iam_admosfera_unam_mx_access.log combined;
       location / {
        proxy_pass               http://127.0.0.1:8080;
        proxy_set_header         X-Real-IP $remote_addr;
        proxy_set_header         X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header         X-Forwarded-Proto https;
        proxy_set_header         Host $http_host;
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        #try_files $uri $uri/ =404;
    }

  listen 443 ssl;      #El servidor escucha peticiones https por el puerto 443.ipv4
  listen [::]:443 ssl; #El servidor escucha peticiones https por el puerto 443,ipv6
# Los protocolos SSL/TLS aceptados para la comunicación segura:
 ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
# Especifica la ubicación del certificado SSL utilizado para la seguridad https:
  ssl_certificate   /etc/letsencrypt/live/iam.atmosfera.unam.mx/cert.pem;
# Especifica la ubicación de la llave privada del certificado SSL:
  ssl_certificate_key  /etc/letsencrypt/live/iam.atmosfera.unam.mx/privkey.pem;
}

#servidor escuchando en el puerto 80
server {
# La condición if verifica si el host solicitado es "iam.atmosfera.unam.mx" entonces
# realiza una redirección a https del mismo host.
       if ($host = iam.atmosfera.unam.mx) {
       return 301 https://$host$request_uri;# return 301 realiza la redirección a
HTTPS
       } # managed by Certbot

    listen 80;      #servidor escucha puerto 80, ipv4
    listen [::]:80; #servidor escucha puerto 80, ipv6

    server_name iam.atmosfera.unam.mx iam;  #nombre del servidor
       return 301 https://$host$request_uri;
}
```

8.3. Archivo de configuración env

```
#Configuración archivo “env” que utiliza el contenedor para levantar Indigo IAM
# The IAM service will list for requests on this host
IAM_HOST=iam.atmosfera.unam.mx
# The IAM service webapp will bind on this port
IAM_PORT=8080
# The IAM web application base URL
IAM_BASE_URL=https://iam.atmosfera.unam.mx/
# The OpenID Connect issuer configured for this IAM instance. This must be equal to
#IAM_BASE_URL
IAM_ISSUER=https://iam.atmosfera.unam.mx/
# The path to the JSON keystore that holds the keys IAM will use to sign and
# verify token signatures.
IAM_KEY_STORE_LOCATION=file:///keystore.jwks
#Los perfiles se habilitan configurando la propiedad del sistema java
#spring.profiles.active. Al iniciar el servicio IAM. Esto se puede hacer, utilizando
#la imagen acoplable oficial de IAM, configurando la variable de entorno IAM_JAVA_OPTS
#de la siguiente manera: IAM_JAVA_OPTS="-#Dspring.profiles.active=prod,oidc,saml"
IAM_JAVA_OPTS=-Dspring.profiles.active=prod,registration
-Djava.security.egd=file:/dev/urandom
IAM_USE_FORWARDED_HEADERS=true
       #Organization configuration
# The name of the organization managed by this IAM instance
IAM_ORGANISATION_NAME="Grid UNAM CU"
# Cadena que se muestra en la barra superior del navegador al acceder al panel de
control de IAM
IAM_TOPBAR_TITLE="UNAM INDIGO IAM for ${IAM_ORGANISATION_NAME}"
       #Data Configuration
# The host where the MariaDB/MySQL daemon is running
IAM_DB_HOST=iam.atmosfera.unam.mx
# The database name
IAM_DB_NAME=iamBD
# The database username
IAM_DB_USERNAME=admin
# The database password
IAM_DB_PASSWORD=#root#
#Notification service settings
# The email address used as the sender in IAM email notifications
IAM_NOTIFICATION_FROM=iam@iam.atmosfera.unam.mx
# The email address used as the recipient in IAM email notifications
IAM_NOTIFICATION_ADMIN_ADDRESS=iam-administrators@iam.atmosfera.unam.mx
# SMTP server hostname
IAM_MAIL_HOST=tlaloc.atmosfera.unam.mx
#El perfil predeterminado de IAM JWT se establece mediante la variable de entorno
#IAM_JWT_DEFAULT_PROFILE
# posibles valores iam, wlcg y aarc
IAM_JWT_DEFAULT_PROFILE=wlcg
```

8.4. Archivo keystore.jwks


```
{
  "keys": [
  {
"p":"1Fw3nvSuxb35e0-jr28f80EOV46HI2ffoNiNF7wDvWESy6dH9AdIancjuogtnjX2jTbl5NlZLVUYYD39C7SnNw",
"kty": "RSA",
"q":"1Dnxh3xhkxjeJ_QlIgqJC86P-MS69LTXLZRpLyLne5_R_MBzUJljlKVf1thu3Z7jgsyDl2svp5WDfFPb1qhb9Q",
"d":"A7LvLdsYxF8nqet9zlHkFNJx_j8bXNxLixlpQebXFA3a-b_E4EFLRVcE2Zvn0NsarYmFTMiXM9GXpnTH3o2xId9esj6w
HxljZCeuaCHdBpU-pu347fCLVevrR47PmGS45Lk_wsIc0nbA6LjwaVTkdsoOECQiNNSPs7rbjbxEdnk",
"e": "AQAB",
"kid": "rsa1",
"qi": "xNX-uae14qnnNJDfZIwFF8d7sPj99iYOYVC1bBaCd4af_Fe6TJ9qBMrJCdvMK31a7ntsaqmKcnVAY7MdCr8g",
"dp": "MGgw-Zf2-vZaF0TJICVpmA6dUVIY9fYooMacKuvdwl6NZUTlfvfmBvbPeyatl2HfqML51pfA3zeFjfKzauasMw",
"dq": "cDqIGG7VJrWy3YwFiv_pQz4j8pkOiNONKwRdr1iU81Ir-K5JDOlguL3Pdc0Uq8wXYhcGy68aD1RIqRmrmtODdQ",
"n":"sAxu8zkwy76Q1sGrMHduAZXPa3mzBtCq7QdRHZP7VSKVxWPzXM0odev09kmDlLbZorWa67cewGFRn8cr7-4YGA-v7kml
oyA6OFM9zPVisKtCN_4gTVZB-6QTQWbgQhoyDqOvTwtJ9VpG_X9ebj-O49OQjiWC_nhueWLZuCFslKM"
  }
        ]
}
```


8.5. Archivo /etc/condor-ce/config.d/10-gridunam.conf

```
#Este archivo contiene los métodos de autenticación disponibles para la grid unam.
SEC_DEFAULT_AUTHENTICATION_METHODS = SCITOKENS,FS,PASSWORD
SEC_CLIENT_AUTHENTICATION_METHODS = SCITOKENS,FS,PASSWORD
SEC_CLIENT_AUTHENTICATION = REQUIRED
SEC_DEFAULT_AUTHENTICATION = REQUIRED

SCHEDD.SEC_READ_AUTHENTICATION_METHODS = $(SEC_DEFAULT_AUTHENTICATION_METHODS)
SCHEDD.SEC_WRITE_AUTHENTICATION_METHODS = $(SEC_DEFAULT_AUTHENTICATION_METHODS)

SCHEDD_DEBUG=D_FULLDEBUG,D_D_ALWAYS,D_SECURITY
```

8.6. Archivo /etc/condor-ce/mapfiles.d/60-gridunam.conf

```
# Este archivo mapea los trabajos entrantes a la cuenta local “usuarioNormal”
# con el método de Autenticación Scitokens.
# SCITOKENS /<TOKEN ISSUER>,<TOKEN SUBJECT>/ <USERNAME>
SCITOKENS /^https:\/\/iam.atmosfera.unam.mx\/,(.*)/ usuarioNormal
SCITOKENS /^https:\/\/grid.atmosfera.unam.mx\/,(.*)/ usuarioNormal
SCITOKENS /^https:\/\/grid.atmosfera.unam.mx\/,0f2b7037-827e-55db-fc2a-f37e203e6444/usuarioNormal
```

8.7. Programa: verToken.sh

```
#!/bin/bash
# Paso 1: Leer el token del usuario y dividirlo en partes
IFS='.' read -ra PARTES <<< "$(read -p "Introduce el token: " -r; echo $REPLY)"

# Paso 2: “Imprimir las partes  del token”
      #Imprimir el Encabezado del token
echo "${PARTES[0]}" | base64 -d | jq
      #Imprimir la Carga Útil del token
echo "${PARTES[1]}" | base64 -d | jq

# Paso 3: Imprimir las fechas: nbf,exp,iat
      echo "FECHAS DEL TOKEN"
   # Usamos útil para almacenar en un string el valor de la Carga Útil del Token
UTIL=$(base64 -d <<< "${PARTES[1]}")

# Dividimos la carga útil del token usando el separador coma ','
IFS=',' read -ra FECHAS <<<  "$UTIL"

# Recorrer e imprimir arreglo FECHAS[ ]
for i in "${FECHAS[@]}"; do
     # Buscar "nbf" en el elemento actual
    if [[ $i == *"\"nbf\":"* ]]; then
        NUMERO_NBF=$(echo "$i" | awk -F '[:,]' '{print $2}')
        echo -e "\nNúmero nbf: Not Before"
      date -d "@$NUMERO_NBF"
    fi

    # Buscar "exp" en el elemento actual
    if [[ $i == *"\"exp\":"* ]]; then
        NUMERO_EXP=$(echo "$i" | awk -F '[:,]' '{print $2}')
        echo -e "\nNúmero exp: Expiration Time"
      date -d "@$NUMERO_EXP"
    fi

    # Buscar "iat" en el elemento actual
    if [[ $i == *"\"iat\":"* ]]; then
        NUMERO_IAT=$(echo "$i" | awk -F '[:,]' '{print $2}')
        echo -e "\nNúmero iat: Issued At"
      date -d "@$NUMERO_IAT"
    fi

done
```

8.8. Comandos útiles

URL donde se pueden encontrar las claves públicas utilizadas para verificar tokens de identidad firmados digitalmente:

```
$ curl "https://grid.atmosfera.unam.mx/.well-known/openid-configuration/" | jq
```

Formas de consultar el certificado del host:

```
root@submit# openssl x509 -in /etc/grid-security/hostcert.pem -text -noout
root@submit# openssl x509 -in /etc/grid-security/hostcert.pem -subject -issuer -dates -noout
```

Consultando versiones de CA, oidc, HTcondor:

```
root@submit#  rpm -qa | grep ca-
root@submit#  rpm -qa | grep oidc
root@submit#  condor_ce_version
```

Verificando el estatus de htcondor-ce. Con este comando podemos verificar los demonios que se están ejecutando en nuestro Htcondor-ce. Esto ofrece pautas para diagnosticar problemas relacionados con los principales demonios: Collector, Scheduler, DaemonMaster, Job_Router.

```
root@submit$  condor_ce_status -any
```

Consultando todos los procesos que se están ejecutando de condor:

```
root@submit#  ps auxwwww | grep condor
```

Verificando la configuración de red en HTCondor-ce.

```
root@submit$  condor_ce_host_network_check
```

Consultando los principales Logs de HTCondorc-ce:

```
root@submit#  tail -f /var/log/condor-ce/MasterLog
root@submit#  tail -f /var/log/condor-ce/CollectorLog
root@submit#  tail -f /var/log/condor-ce/SchedLog
```

Consultando los métodos de autenticación del Collector, Schedd. La variable SEC_DEFAULT_AUTHENTICATION_METHODS lista de forma ordenada los métodos de autenticación permitidos. Estos métodos se prueban en orden hasta que uno tiene éxito o todos fallan.

```
root@submit $ condor_ce_config_val -v COLLECTOR.SEC_ADVERTISE_STARTD_AUTHENTICATION_METHODS
root@submit $ condor_ce_config_val -v SCHEDD.SEC_READ_AUTHENTICATION_METHODS
SCHEDD.SEC_WRITE_AUTHENTICATION_METHODS
root@submit $ condor_ce_config_val SEC_DEFAULT_AUTHENTICATION_METHODS
```

Consultando la variable SEC_DEFAULT_ENCRYPTION, esta variable puede tener los siguientes valores:REQUIRED, PREFERRED, OPTIONAL, NEVER. Nuestro valor es: REQUIRED. Esto indica que cualquier comando que requiera autorización de ESCRITURA fallará a menos que el canal esté cifrado.

```
root@submit# condor_ce_config_val   SEC_DEFAULT_ENCRYPTION
```

Exportando la variable Debug, la cual nos ayudará al momento de depurar posibles errores en nuestra configuración.

```
root@submit$  export  _condor_TOOL_DEBUG=D_FULLDEBUG,D_D_ALWAYS,D_SECURITY
```

Debug: se utiliza condor_ce_ping con la opción debug para habilitar la visualización de información adicional de depuración durante la ejecución del comando.

```
root@submit$  condor_ce_ping -debug
```

Verbose: se utiliza condor_ce_ping con la opción -verbose la cual nos muestra información adicional sobre la autenticación.

```
root@submit$ condor_ce_ping -verbose
```

Este comando nos permite visualizar el recorrido completo del trabajo y así probar el envío de extremo a extremo en un entorno HTCondor-CE.

```
root@submit$  condor_ce_trace --debug submit.atmosfera.unam.mx
```

La opción -dump para condor_config_val imprime los valores de las variables de configuración que coinciden con un patrón dado. Si no se especifica un patrón, se imprimirán todos los valores de configuración disponibles. Por ejemplo:

```
root@submit$ condor_ce_config_val -dump SSL
```

Esta opción no mostrará todo lo relacionado con SSL de nuestra configuración htcondor-ce.

BIBLIOGRAFÍA

1. HTCondor Software Suite, *HTCondor 8.9.2 Released* [en línea] [consulta: 01 junio 2024]. Disponible en: <https://research-cs-wisc-edu.translate.goog/htcondor/news/HTCondor_8.9.2_released!/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc>.
2. González. A., *Comisión Técnica de la Grid UNAM aprueba documentos base de normatividad*. 2022. [en línea] [Consulta: 10 marzo 2022]. Disponible en: <https://www.tic.unam.mx/comision-tecnica-de-la-grid-unam-aprueba-documentos-base-de-normatividad/>
3. Guerrero Estrada, J. M. y Jimenez Bacca, D. E., Computación Grid. Universidad Tecnológica de Bolívar. 2011. 72p.
4. Zuluaga Moreno, I. C., Giraldo Escobar, J. O. & Gómez Castaño J. C., *Redes Académicas de Alta Calidad.* Centro de Investigaciones y Desarrollo. Facultad de Ciencias e Ingeniería., Diciembre 2010.
5. U. S. Departament of Energy. *Acerca de ESnet.* [en línea] [Consulta: 23 abril 2023]. Disponible en: <https://my.es.net/>
6. RedCLARA, *ALICE.* [en línea] [Consulta: 11 marzo 2022]. Disponible en: <https://redclara.net/es/colaboracion/proyectos/red-e-infraestructura/alice>
7. RedCLARA, *ALICE 2.* [en línea] [Consulta: 11 marzo 2022]. Disponible en: <https://www.redclara.net/es/colaboracion/proyectos/red-e-infraestructura/alice2>
8. RedCLARA,*The BELLA Consortium.*[en línea] [Consulta: 12 marzo 2022]. Disponible en: <https://bella-programme.redclara.net/index.php/en/about-bella/the-bella-programme/the-bella-consortium>
9. Corporación Universitaria para el Desarrollo de Internet.(s.f.). *CUDI. Corporación Universitaria para el Desarrollo de Internet.* [en línea] [Consulta: 14 Enero 2023]. Disponible en: <https://cudi.edu.mx/sites/default/files/flmngr/infografia_CUDI.pdf>
10. E - infrastructure Shared Between Europe and Latin America. *ABOUT EELA.* [en línea] [Consulta: 01 junio 2024]. Disponible en: <https://www.eu-eela.eu/first-phase.php>
11. RedCLARA, *ELLA 2.*[en línea] [Consulta: 01 junio 2024]. Disponible en: <https://redclara.net/es/colaboracion/proyectos/mallas-computacionales/eela-2>
12. U.S. Departament of Energy. *Our Mission.*[en línea] [Consulta: 10 mayo 2023]. Disponible en: <https://www.es.net/about/our-mission/>
13. RedCLARA, *GISELA.* [en línea] [Consulta: 01 junio 2024]. Disponible en: <https://redclara.net/es/colaboracion/proyectos/mallas-computacionales/gisela>
14. FIREANT STUDIO, *About Us Internet2.* [en línea] [Consulta: 01 junio 2024]. Disponible en:<https://internet2.edu/community/about-us/>
15. Tecnológico Nacional de México. *¿Qué es Internet 2?*.[en línea] [Consulta: 02 abril 2022]. Disponible en: <http://www.dgest.gob.mx/telecomunicaciones/que-es-internet-2>

16. Smallen s. & Shimojo. S. *Pacific Rim Application and Grid Middleware Assembly.* Collaborative Overview. (20):3-4, 2019. <https://www.pragma-grid.net/images/collab/CollaborativeOverview2019.pdf>
17. RedCLARA,*Topología Actual de la Red.* [en línea] [Consulta: 01 junio 2024]. Disponible en: <https://www.redclara.net/index.php/es/red/redclara/topologia-actual-de-la-re>
18. RedCLARA, *Remote Instrumentation on Next Generation Grids - RINGrid.* [en línea] [Consulta: 01 junio 2024]. Disponible en: <https://redclara.net/es/colaboracion/proyectos/mallas-computacionales/ringrid>
19. Sciauth. (s.f.). *Proyect Summary*. Universidad Illinois. [en línea] [Consulta: 01 mayo 2022]. Disponible en: <https://sciauth.org/sciauth-proposal-public.pdf>
20. X.509. Tecnología de la información- Interconexión de sistemas abiertos - El directorio: Marcos para certificados de claves públicas y atributos. Ginebra. 139 p.
21. Microsoft. *Certificados X.509.* [en línea] [Consulta: 01 marzo 2023]. Disponible en: <https://learn.microsoft.com/es-es/azure/iot-hub/reference-x509-certificates>
22. Microsoft. *Autenticación de identidades con certificados X.509.* [en línea] [Consulta: 27 Febrero 2023]. Disponible en: <https://learn.microsoft.com/es-es/azure/iot-hub/iot-hub-x509-certificate-concepts>
23. Equipo de soporte de SSL.com. *¿Qué es un certificado X 509?* [en línea] [Consulta: 12 marzo 2023]. Disponible en: <https://www.ssl.com/es/preguntas-frecuentes/%C2%BFQu%C3%A9-es-un-certificado-x-509%3F/>
24. Kaspersky. *Qué es un certificado SSL: Definición y Explicación.* [en línea] [Consulta: 01 junio 2024]. Disponible en: <https://latam.kaspersky.com/resource-center/definitions/what-is-a-ssl-certificate>
25. Jones, M., Bradley, J., & Sakimura. N. (Mayo 2015). *JSON web Token (JWT).* RFC 7519. Internet Engineering Task Force (IETF). [en línea] [Consulta: 06 junio 2023]. Disponible en: <https://www.rfc-editor.org/rfc/rfc7519.html>
26. Peyrott. S. JWT HANDBOOK. version 0.14.1. Auth0 Inc. (2016 - 2018) .119 p.
27. Buthmann, B. *European Identity & Cloud Awards 2014.* (15 Mayo 2014).[en línea] [Consulta: 26 junio 2023]. Disponible en: <https://www.kuppingercole.com/blog/buthmann/award2014>
28. Flores Salgado. L. Y. Configuración y puesta en operación de un Clúster HPC para la Unidad MOFABI de la Facultad de Ingeniería de la UNAM. Tesis (Especialista en cómputo de alto rendimiento). CD. MX., Universidad Autónoma de México. Noviembre 2018. 47 p
29. Blaise. B., Frederic. D. *Introduction to Parallel Computing Tutorial.* [en línea] [Consulta: 23 Abril 2023]. Disponible en: <https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial##Overview>
30. OSPool Documentación, *Computation on the Open Science* Pool. [en línea] [Consulta: 02 Agosto 2022]. Disponible en: <https://portal.osg-htc.org/documentation/overview/account_setup/is-it-for-you/>

31. Partnership to Advance Throughput Computing. (2021, Agosto, 02) *Introduction to the Virtual School, HTC, and OSG.* Youtube. <https://www.youtube.com/embed/vpJPPjoQ3QU>
32. Thain. D. Tannenbaum. T. & Livny. M. *Distributed Computing in Practice: The Condor Experience.*(2004) University of Wisconsin-Madison. [Consulta: 28 Agosto 2022].
33. HTCondor Software Suite, *HTCondor OverView.* [en línea] [Consulta: 23 Abril 2023]. Disponible en :<https://htcondor.org/htcondor/overview/>
34. Center for High Throughput Computing, *Administrative Quick Start Guide.* [en línea] [Consulta: 27 febrero 2024]. Disponible en: <https://htcondor.readthedocs.io/en/latest/getting-htcondor/admin-quick-start.html?highlight=execute#the-execute-role>
35. Koch, Christina., *2022-HTCondor-User-Tutorial.pdf.*[en línea] [Consulta: 16 diciembre 2023]. Disponible en: <https://htcondor.org/event_summary/htcondor_week_2022.html>
36. Center for High Throughput Computing, *Security. Resumen de seguridad.* [en línea] [Consulta: 10 noviembre 2023]. Disponible en: <https://htcondor.readthedocs.io/en/latest/admin-manual/security.html#password-authentication>
37. HTCondor-CE Documentation. *Installing HTCondor-CE 23.* [en línea] [Consulta: 23 abril 2023]. Disponible en: <https://htcondor.com/htcondor-ce/v23/installation/htcondor-ce/>
38. Hua Lin. B. *HTCondor-CE Basics and Architecture.* [en línea] Roma, 26 septiembre 2019, [Consulta: 20 octubre 2023]. Disponible en: <https://indico.cern.ch/event/817927/contributions/3570555/>
39. SchedMD.*Quick Start User Guide.*[en línea] 29 Junio 2021, [Consulta: 25 julio 2023]. Disponible en: <https://slurm.schedmd.com/quickstart.html>
40. Bertocci, V. & Chiarelli, A. *OAuth2 and OpenID Connect: The Professional Guide - BETA.* Auth0. 109 p.
41. Hammer - Lahv. E. (Abril 2010). *The OAuth 1.0 Protocol.* RFC 5849. Internet Engineering Task Force (IETF). [en línea] [Consulta: 06 junio 2023]. Disponible en: <https://www.rfc-editor.org/rfc/rfc5849>
42. Hardt. D. (Octubre 2012).*The OAuth 2.0 Authorization Framework.* RFC 6749. Internet Engineering Task Force (IETF). [en línea] [Consulta: 06 junio 2023]. Disponible en: <https://www.rfc-editor.org/rfc/rfc6749#section-1.1>
43. Sakimura. N., Bradley. J., Jones B. M., De Medeiros. B. & Mortimore. M.(15 diciembre 2023). *OpenID Connect Core 1.0 incorporating errata set 2.* [en línea] [Consulta: 08 junio 2023]. Disponible en: <https://openid.net/specs/openid-connect-core-1_0.html>
44. Git Book. *Introduction*. [en línea] [Consulta: 23 abril 2023]. Disponible en: <https://indigo-dc.gitbook.io/iam/>
45. INFN. *OverView: A brief overview of the INDIGO IAM service*. [en línea] (09 Septiembre, 2021)[Consulta: 14 julio 2023].Disponible en: <https://indigo-iam.github.io/v/v1.8.2/docs/overview/>
46. Hernandez Cervantes. L., Santillan González. A. J. & Caballero Cruz. R. E. *Maestros y Esclavos una Aproximación a los Cúmulos de Computadoras*. [en línea] [Consulta: 08 junio 2023]. Disponible en: <https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.r

evista.unam.mx/vol.4/num2/art3/jun_art3.pdf&ved=2ahUKEwj3_6yH7ruGAxUZC0QIHbZVAUEQFnoECBcQAQ&usg=AOvVaw09N4OHJWyucDLo0ZGMHgWF >
47. SchedMD. *Slurm version 24.05. Configuration Tool*. [en línea] (13 marzo 2024) [Consulta: 01 junio 2024]. Disponible en: <https://slurm.schedmd.com/configurator.html>
48. IBM,*Inicio de sesión Único (SSO).* [en línea] [Consulta: 01 junio 2024]. Disponible en: <https://www.ibm.com/es-es/topics/single-sign-on>
49. Hardt, D. (Octubre 2012). The OAuth 2.0. Authorization Framework. RFC 6749. Internet Engineering Task Force (IETF).[en línea] [Consulta: 01 junio 2024]. Disponible en: <https://datatracker.ietf.org/doc/html/rfc6749>
50. Denniss, W., Bradley. J., Jones. M. B., & Tschofenig. H. (Agosto 2019). *OAuth 2.0 Device Authorization Grant.* RFC 8628. Internet Engineering Task Force (IETF).[en línea] [Consulta: 01 junio 2024]. Disponible en: <https://www.rfc-editor.org/rfc/rfc8628#section-1>
51. MkDocs, *HTC Exercise 1.1: Log In and Look Around.* [en línea] (6 Agosto 2023) [Consulta: 02 Febrero 2024]. Disponible en: <https://osg-htc.org/user-school-2023/materials/htcondor/part1-ex1-login/>
52. HTCondor-CE Documentation. *Configuring Authentication*. [en línea] [Consulta: 22 febrero 2024].Disponible en: <https://htcondor.com/htcondor-ce/v6/configuration/authentication/>
53. MkDocs, *HTC Exercise 2.4:Submit With "queue matching"* [en línea] (6 Agosto 2023). [Consulta: 15 Febrero 2024]. Disponible en: <https://osg-htc.org/user-school-2023/materials/htcondor/part2-ex4-queue-matching/>