Abner Velasco

dominion global

 Ibáñez de Bilbao, 28 8º A y B48009 BILBAO

jitsi meet aws eks



Tabla de contenido

[1. Jitsi Meet 2](#_Toc53366968)

[1.1. Arquitectura 2](#_Toc53366969)

[1.1.1. Imágenes 3](#_Toc53366970)

[1.1.2. Puertos necesarios 4](#_Toc53366971)

[1.1.3. XMPP 4](#_Toc53366972)

[1.1.4. Edición de imágenes Docker 4](#_Toc53366973)

[2. Instalación 4](#_Toc53366974)

[2.1. Por qué utilizar Kustomize y no Helm 6](#_Toc53366975)

[2.2. Archivos de instalación 6](#_Toc53366976)

[2.3. Extructura del proyecto 7](#_Toc53366977)

# Jitsi Meet

Jitsi es una [colección de proyectos de código abierto](https://jitsi.github.io/handbook/docs/architecture) que proporcionan capacidades de videoconferencia de última generación que son seguras, fáciles de usar.

## Arquitectura

Nos basaremos en la versión autohospedada de Jitsi meet, la cual indica los siguientes pasos para poder obtener una instancia en docker.

* Descargue y extraiga la [última versión](https://github.com/jitsi/docker-jitsi-meet/releases/latest)
* Alternativamente, para probar los últimos cambios clone el repositorio: git clone https://github.com/jitsi/docker-jitsi-meet && cd docker-jitsi-meet
* Cree un archivo copiando y ajustando .envenv.example
* cp env.example .env
* Establezca contraseñas seguras en las opciones de la sección de seguridad del archivo ejecutando el siguiente script bash .env
* ./gen-passwords.sh
* Crear directorios necesarios CONFIG
* mkdir -p ~/.jitsi-meet-cfg/{web/letsencrypt,transcripts,prosody/config,prosody/prosody-plugins-custom,jicofo,jvb,jigasi,jibri}
* Para Windows: echo web/letsencrypt,transcripts,prosody/config,prosody/prosody-plugins-custom,jicofo,jvb,jigasi,jibri | % { mkdir "~/.jitsi-meet-cfg/$\_" }
* Ejecutar docker-compose up -d
* Acceda a la interfaz de usuario web en [https://localhost:8443](https://localhost:8443/)(o a un puerto diferente, en caso de que haya editado el archivo de composición).

Tenga en cuenta que HTTP (no HTTPS) también está disponible (en el puerto 8000, de forma predeterminada), pero eso es, por ejemplo, para una configuración de proxy inverso; El acceso directo a través de HTTP en su lugar HTTPS conduce a errores WebRTC como:

No se pudo acceder a su micrófono/cámara:

No se puede utilizar el micrófono/cámara por una razón desconocida.

No se puede leer la propiedad 'getUserMedia' de undefined o navigator.mediaDevices es undefined.

Si también desea utilizar jigasi, primero configure el archivo env con credenciales SIP y, a continuación, ejecute Docker Compose de la siguiente manera: docker-compose -f docker-compose.yml -f jigasi.yml up

Si desea habilitar el uso compartido de documentos a través de [Etherpad](https://github.com/ether/etherpad-lite), configúrelo y ejecute Docker Compose de la siguiente manera: docker-compose -f docker-compose.yml -f etherpad.yml up

Esta configuración solía tener contraseñas predeterminadas para las cuentas internas utilizadas en todos los componentes. Para hacer la configuración predeterminada segura por abandono estos se han quitado y los contenedores respectivos no se iniciarán sin tener una contraseña establecida.

Las contraseñas seguras se pueden generar de la siguiente manera: Esto modificará su archivo (una copia de seguridad se guarda en ) y establecerá contraseñas seguras para cada una de las opciones requeridas. Las contraseñas se generan mediante ../gen-passwords.sh.env.env.bakopenssl rand -hex 16

NO reutilice ninguna de las contraseñas.

Una instalación de Jitsi Meet se puede dividir en los siguientes componentes:

* Una interfaz web
* Un servidor XMPP
* Un componente de enfoque de conferencia
* Un router de vídeo (podría ser más de uno)
* Una puerta de enlace SIP para llamadas de audio
* Una infraestructura de radiodifusión para grabar o transmitir una conferencia.

Diagrama

Descripción generada automáticamente

## Imágenes

* **base**: Imagen base estable de Debian con la [superposición S6](https://github.com/just-containers/s6-overlay) para el control de procesos y los [repositorios Jitsi habilitados.](https://jitsi.org/downloads/) Todas las demás imágenes se basan en esta.
* **base-java**: Igual que la anterior, más Java (OpenJDK).
* **web**: Jitsi Meet web UI, servido con nginx.
* **prosody**: [Prosody,](https://prosody.im/) el servidor XMPP.
* **jicofo**: [Jicofo](https://github.com/jitsi/jicofo), el componente de enfoque XMPP.
* **jvb**: [Jitsi Videobridge](https://github.com/jitsi/jitsi-videobridge), el router de vídeo.
* **jigasi**: [Jigasi](https://github.com/jitsi/jigasi), la puerta de enlace SIP (solo audio).
* **etherpad**: [Etherpad](https://github.com/ether/etherpad-lite), complemento de edición de documentos compartidos.
* **jibri**: [Jibri](https://github.com/jitsi/jibri), la infraestructura de radiodifusión.

## Puertos necesarios

* 80/tcp para Web UI HTTP
* 8080/tcp para API Rest de estadistica
* 443/tcp para web UI HTTPS
* 4443/tcp para medios RTP a través de TCP
* 10000/udp para medios RTP sobre UDP

## XMPP

Jitsi Meet utiliza XMPP para la señalización, por lo tanto la necesidad del servidor XMPP. La configuración proporcionada por estos contenedores no expone el servidor XMPP al mundo exterior. En su lugar, se mantiene completamente sellado, y el ruteo del tráfico XMPP ocurre solamente de manera interna.

## Edición de imágenes Docker

La creación de imágenes le permite editar los archivos de configuración de cada imagen individualmente, lo que proporciona más personalización para la implementación.

Las imágenes acoplables se pueden crear ejecutando el comando en la carpeta principal del repositorio. Si necesita sobrescribir imágenes existentes desde el origen remoto, utilice .makeFORCE\_REBUILD=1 make

Si se encuentra en la rama inestable, cree las imágenes con .FORCE\_REBUILD=1 JITSI\_RELEASE=unstable make

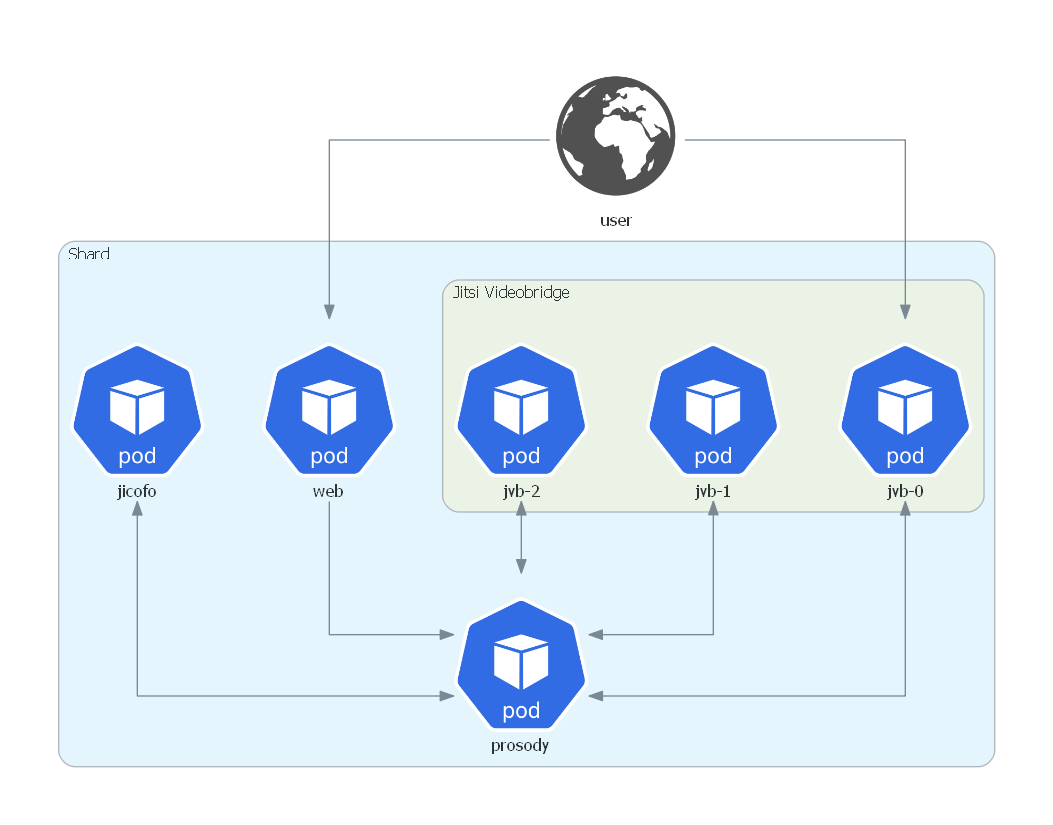
Para mas información consultar la documentación proporcionada por Kelvin

# Instalación

La instalación se realizó en un cluster EKS de Amazon por lo tanto todos los archivos presentados a continuación están optimizados para dicho escenario, esto posiblemente no funcionará bajo otras circunstancias. Requerimientos:

* Cluster de Kubernetes en su versión 1.17 o superior
* VPC creado por default
* 2 nodos los cuales tienen que ser configurados en maquinas t3.xlarge
* Dominio
* Certificado SSL firmando el Balanceador de carga

A continuación un esquema demostrando la arquitectura en un cluster de kubernetes eks en aws.



La gráfica anterior muestra cada una de las imágenes docker mensionadas en el punto 1.1.1. Para un escalado exitoso en el cluster de kubernetes se debe de cumplir las siguientes condiciones:

* Un solo pod para la imagen web
* Un solo pod para la imagen jicofo
* Un solo pod para la imagen prosody
* Varios pod para la imagen de jvb

La imagen jvb es la que permitirá poder escalar en numero de participantes en una o varias video conferencias, esto es lo que determina la cantidad de usuarios que pueden utilizar el servicio.

Notará la necesidad de incrementar una replica del pod de jvb cuando el video se deteriore de forma notoria para todos los usuarios.

Las imágenes que solamente tienen un pod (web, jicofo, prosody) tambien pueden escalarse, un factor notorio para la toma de decisión de este escenario será cuando la conexión a una sala y el transcurrrir en ella (que no tenga que ver con video conferencia) resulte de forma lenta.

Servirá mucho el monitoreo del cluster para determinar que las imágenes puestas en los pods no lleguen a su límite de memoria.

Para poder levantar el servicio de jitsi utilizamos Kustomize que es un cliente que nos permite utilizar yaml puro.

## Por qué utilizar Kustomize y no Helm

La mejor manera de describir las diferencias es hacer referencia a ellas como diferentes tipos de motores de implementación. Uno es un *motor de plantillas* y otro es un motor de *superposición.*

Entonces, ¿qué son estos? Bueno, cuando se utiliza un motor de plantillas se crea un ejemplo reutilizable de su archivo. A partir de ahí se abstrae el contenido con filtros conocidos y dentro de estas abstracciones se proporcionan referencias a variables. Estas variables normalmente se abstraen a otro archivo donde se inserta información específica de su entorno A continuación, en tiempo de ejecución, cuando se ejecuta el motor de plantillas, las plantillas se cargan en la memoria y todas las variables se intercambian con sus marcadores de posición.

Esto es diferente de un motor de superposición de algunas maneras matizadas. Normalmente sobre cómo la información entra en los ejemplos de configuración. Kustomize no usa plantillas. En su lugar, cree un archivo *Kustomization.yml.* Este archivo entonces apunta a dos cosas diferentes. Tu *Base* y tus *Superposiciones*. En tiempo de ejecución, la Base se carga en la memoria y, si existen superposiciones, se combinan encima de la configuración base.

Puede hacer referencia a bases de Internet u otras rutas no estándar.

Es compatible con generadores para crear archivos de configuración automáticamente en función de archivos y literales de cadena.

Admite parches JSON robustos y granulares.

Admite la inserción de metadatos entre archivos de configuración.

## Archivos de instalación

En nuestra extensa investigación y periodo de prueba, ensayo y error, nos dimos cuenta que necesitabamos más control conrespecto a los componentes que se instalan y tambien instalar otros componentes propios de AWS como lo son un Ingress y un balanceador de carga.

A este documento lo acompaña un archivo zip con el nombre de Project (Project.zip) el cual contiene todos los archivos necesarios para el despliegue de Jitsi en un EKS de AWS. El proyecto crea lo siguiente:

* Balanceador de carga configurado con certificado SSL autofirmado
* Ingress para el Balanceador
* LoadBalancer para el Balanceador
* Namespace
* Imagen Web
* Imagen Jicofo
* Imagen Prosody
* Imagen JVB
* Configuración necesaria para AWS
* Herramientas de monitoreo (Opcional)

## Extructura del proyecto

Imagen que contiene Diagrama

Descripción generada automáticamente