

# AUTOAPROVISIONAMIENTO VDI

## Descripción del servicio de auto promoción de escritorios virtuales

UOC

### Índice

<b>Objetivo</b>	<b>2</b>
<b>Visión global</b>	<b>2</b>
<b>Control de usuarios (autenticación y autorización)</b>	<b>3</b>
<b>Frontal de aprovisionamiento</b>	<b>7</b>
2.1. Usuario autorizado a crear	10
2.2. En creación	10
2.3. Creado	10
Método principal	11
Método para obtener el estado	12
Configuración de los endpoints	12
<b>API Workspaces</b>	<b>12</b>
<b>Funciones Lambda</b>	<b>13</b>
Función de provisión	13
Función de comprobación	15
Repositorio Github	16
<b>VPN y redes</b>	<b>16</b>
Creación de servicio de VPC en AWS	16
Creación del servicio VPN Site-to-Site	17
<b>Conexión con AD</b>	<b>18</b>
Security Group de los WORKSPACES	20
<b>Imagen plantilla y promoción</b>	<b>21</b>
Instalación inicial del VDI "maqueta"	21
Promoción de la maqueta	21
Restauración de un VDI	22

## Objetivo

El objetivo es poder ampliar los mecanismos de teletrabajo de la UOC proporcionando entornos virtuales con acceso a las aplicaciones corporativas para ciertos usuarios cuando la situación lo requiera.

## Visión global

En este documento se describe el sistema de aprovisionamiento de escritorios virtuales (VDI) en AWS Workspaces que se ha puesto en marcha en la UOC para permitir el teletrabajo. El funcionamiento general es el siguiente:

1. Un usuario entra en una página web que permite el aprovisionamiento. Esta página es capaz de comprobar si el usuario tiene permisos (ver más adelante), y en caso afirmativo si dispone ya de un escritorio virtual. En caso de que no esté creado le permitirá solicitarlo, y en caso de que ya esté creado le dará información de cómo acceder. Se permite entrar sólo a usuarios autenticados en el portal y que estén permitidos bien por lista de usuarios nominales, bien por rol (afiliación).
2. En caso de que el usuario solicite la creación de un escritorio virtual, se pide la creación mediante la API a AWS y se le pide al usuario que espere. Mientras se usa la API de comprobación del estado de la creación para indicar al usuario cuándo ha sido creada. La creación de la máquina hace todas las tareas necesarias con el Active Directory, tales como el registro en el dominio.
3. Cuando el escritorio virtual ya ha sido creado, se le informa al usuario del modo de acceso con la URL de descarga del cliente y el token necesario para entrar.
4. En la entrada al escritorio virtual, que es la VDI creada para la UOC, se le piden al usuario las credenciales de acceso al Active Directory corporativo para poder activar el escritorio virtual. Este escritorio ha sido configurado de tal modo que tiene acceso a todas las herramientas y unidades corporativas, es decir, como si fuera un PC más dentro de la red interna.
5. Cuando el usuario deja de usar el escritorio durante un tiempo determinado, la máquina se apaga automáticamente.

## Control de usuarios (autenticación y autorización)

La autenticación en el entorno UOC se hace vía SAML en el proveedor de Identidad de la UOC. El proceso de autenticación se lanza desde la página inicial de nuestro portal (<https://www.uoc.edu>) y cuando tiene éxito genera un JWT copiando los valores que recibe en el SAML response (por tanto con los datos del usuario), que se inyecta como cookie (campusJWT).

Un ejemplo del contenido descriptado del JWT es:

```
{ "alg": "HS256" } { "jti": "SD0cKpeFkK", "iat": 1584525280, "sub": "{
  \"schacPersonalUniqueCode\" :
  \"urn:mace:terena.org:schac:personalUniqueCode:es:rediris:sir:mbid:{sha1
  }cfded6fe8f2d57e4efdf3d9724d48affcc2b5f66\",
  \"eduPersonEntitlement\" :
  \"urn:mace:dir:entitlement:common-lib-terms\",
  \"commonName\" : \"NOMBRE Y APELLIDOS\",
  \"preferredLanguage\" : \"ca\",
  \"mail\" : \"LOGIN@uoc.edu\",
  \"eduPersonAffiliation\" : \"student\",
  \"displayName\" : \"NOMBRE Y APELLIDOS\",
  \"givenName\" : \"NOMBRE\",
  \"schacPersonalUniqueID\" : \"urn:schac:personalUniqueID:es:NIF:DNI\",
  \"schacHomeOrganizationType\" :
  \"urn:mace:terena.org:schac:homeOrganizationType:int:university\",
  \"schacGender\" : \"M\",
  \"employeeNumber\" : \"EMPLOYEEENUMBER\",
  \"uid\" : \"LOGIN\",
  \"eduPersonPrimaryAffiliation\" : \"staff\",
  \"eduPersonScopedAffiliation\" : \"staff@uoc.edu\",
  \"schacHomeOrganization\" : \"uoc.edu\",
  \"eduPersonPrincipalName\" : \"EMPLOYEEENUMBER@uoc.edu\",
  \"sn\" : \"APELLIDOS\"
}", "iss": "UOC", "exp": 1584611680 }
```

Para autenticar al usuario se ha creado una función serverless en AWS (un Lambda en NodeJS) que valida el JWT y comprueba si debe dejar pasar o no al usuario. La función está preparada para funcionar de dos modos: o bien dejar pasar a usuarios con un rol (`eduPersonPrimaryAffiliation`) determinado o bien a los usuarios de una lista de logins. El lambda tiene el siguiente aspecto (sólo partes relevantes):



```

const jwt = require('jsonwebtoken');
const auth_users = require('./auth_users');
const buildIAMPolicy = (userId, effect, resource, context) => {
  const policy = {
    principalId: userId,
    policyDocument: {
      Version: '2012-10-17',
      Statement: [
        {
          Action: 'execute-api:Invoke',
          Effect: effect,
          Resource: resource,
        },
      ],
    },
    context,
  };

  return policy;
};

module.exports.handler = (event, context, callback) => {
  const token = event.authorizationToken;
  const auth_mode = process.env.auth_mode || "role"; // Funcionar con rol o lista usuarios

  try {
    // Verify JWT
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    const sub = JSON.parse(decoded.sub);
    const user = sub.uid;
    const authorizerContext = { user: JSON.stringify(user) };

    let policyDocument = null;

    const valid_affiliations = process.env.affiliations ? process.env.affiliations.split(",") :
["staff"];

    if((auth_mode==="role" && valid_affiliations.indexOf(sub.eduPersonPrimaryAffiliation)===-1) ||
(auth_mode==="user" && auth_users.indexOf(user)===-1)){
      policyDocument = buildIAMPolicy(user, "Deny", event.methodArn, authorizerContext);
      callback(null, policyDocument);
    }

    // Return an IAM policy document for the current endpoint
    policyDocument = buildIAMPolicy(user, "Allow", event.methodArn, authorizerContext);
    callback(null, policyDocument);

  } catch (e) {
    callback(e); // Return a 401 Unauthorized response
  }
};

```

Donde hay definidas una serie de variables que son:

- **JWT\_SECRET** → secret usado para la JWT
- **auth\_mode** → “user” (lista de usuarios) o “role” (por rol)
- **affiliations** → afiliaciones que podrán entrar separadas por comas (p.ej. “staff,employee”)

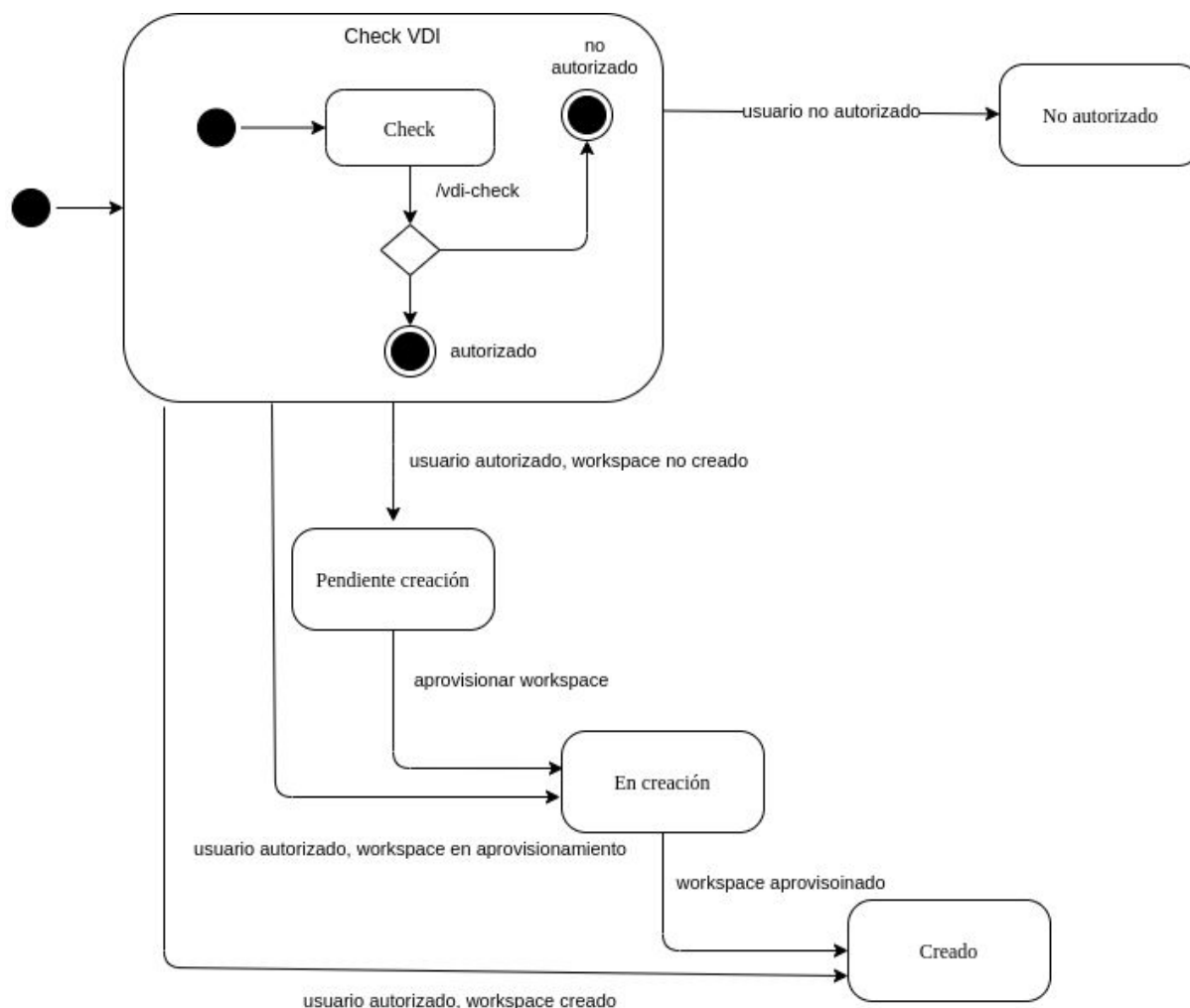
Además existe un fichero `auth_users.js` con la lista de usuarios, de este modo:

```
module.exports = [  
  "login1",  
  "login2",  
  (...)  
];
```

## Frontal de aprovisionamiento

En la intranet de la UOC, denominada IntraUOC, se ha habilitado una nueva página web donde se expone al personal UOC las diferentes opciones de teletrabajo disponibles incluyendo la nueva basada en AWS Workspaces.

El diagrama de estados de la página es el siguiente:



En el momento de cargar la página se invoca a la API “vdi-check” para saber si el usuario está autorizado o no a usar AWS Workspaces. En caso afirmativo, además se obtiene información del estado en el que se encuentra su aprovisionamiento.

En función del resultado, el usuario visualizará un contenido diferente.

### 1) **No autorizado** (Usuario sin permisos en AWS Workspaces)

En caso de no estar autorizado, se le muestran las opciones disponibles de teletrabajo que no son AWS Workspaces.

## Mecanismes de teletreball

Prioritza l'ús de la VPN per sobre de l'ús del Citrix (Teletreball).  
A la IntraUOC hi ha una llista bàsica de les eines accessibles des d'un sistema i l'altre.

[Instruccions per a connectar al Teletreball Citrix](#)  
[Client VPN Pulse \(Windows\)](#)  
[Client VPN Pulse \(Mac\)](#)

### 2) Autorizado (Usuario con permisos en AWS Workspaces)

En caso de estar autorizado, se le proporciona información de cómo aprovisionar el entorno de teletrabajo en AWS Workspaces.

#### Sistema complementari al Citrix (Teletreball)

Per a facilitar l'e-treball, l'Àrea de Tecnologia ha habilitat un sistema complementari al Citrix (Teletreball) per a alguns usuaris. Es tracta d'un sistema d'aprovisionament d'escriptoris en remot. Aquest nou sistema, amb un sol clic, et permetrà disposar d'un escriptori en remot amb accés als recursos interns de la UOC. Tots aquells usuaris de teletreball amb necessitats específiques, com ara accés a unes aplicacions determinades, caldrà que continuïn utilitzant Citrix. La resta d'usuaris podran habilitar aquest nou sistema d'escriptoris en remot.

#### Instruccions

En primer lloc cal que cliquis el botó que trobaràs a la secció de més avall, com el següent

Crea el teu escriptori remot

Veuràs en pantalla que et surt aquest missatge. L'escriptori pot trigar fins a 20 minuts a crear-se. Pots marxar d'aquesta pàgina i tornar al cap d'una estona o navegar per altres pestanyes del navegador, si et convé.

El teu escriptori està creant-se. Pots esperar o tornar a aquesta pàgina en uns minuts



Veuràs en pantalla un missatge que et donarà unes instruccions:

1. El teu escriptori ja està creat
2. Utilitza el codi de registre següent: xxxxxxxxxxxx
3. Has de connectar a <https://clients.amazonworkspaces.com/>
4. Descarrega el client que millor s'adapti a la teva plataforma (Windows, Linux, Mac)
5. Per accedir a l'escriptori has d'utilitzar les teves credencials d'accés a Campus (sense @uoc.edu)



## 2.1. Usuario autorizado a crear

Crea el teu escriptori remot

La ejecución del evento de creación del escritorio remoto invoca a la API “**vdiprovision**”.

## 2.2. En creación

El teu escriptori està creant-se. Pots esperar o tornar a aquesta pàgina en uns minuts



## 2.3. Creado

1. El teu escriptori ja està creat
2. Utilitza el codi de registre següent: 
3. Has de connectar a <https://clients.amazonworkspaces.com/>
4. Descarrega el client que millor s'adapti a la teva plataforma (Windows, Linux, Mac)
5. Per accedir a l'escriptori has d'utilitzar les teves credencials d'accés a Campus (sense @uoc.edu)

El **código de registro** debe ser suministrado por el administrador de AWS Workspaces de la organización.

A continuación se muestra algún "snippet" de código Javascript:

## Método principal

```

async function main(){
  try{
    let state = await getState();
    if(!state){
      vdiButton.style.display = "block";
    }else{
      switch(state){
        case "TERMINATING":
          state="ERROR";
          break;
        case "STOPPED":
          state="AVAILABLE";
          break;
        case "PENDING":
          reload();
          break;
        case "UNAUTHORIZED": //TT NORMAL, NO AWS
          console.log("No autoritzat");
          vdiLoading.style.display = "none";
          vdiRegular.style.display = "inline-block";
          return;
        }
      }
      const el = document.getElementById("vdi-info-"+ state);
      if(el){
        el.style.display = "block";
      }
    }
    vdiWidget.style.display = "inline-block";
    vdiLoading.style.display = "none";
  }catch(e){
    console.error(e);
  }
}

```

## Método para obtener el estado

```

async function getState(){
  let state = null;
  let workspaces = null;

```

```
try{
  let response = await request(url+checkEndpoint, jwt, "POST");
  workspaces = JSON.parse(response);
}catch(e){
  console.log("UNAUTHORIZED");
  state="UNAUTHORIZED";
}

if(workspaces && workspaces.Workspaces &&
workspaces.Workspaces.length>0){
  state = workspaces.Workspaces[0].State;
}
return state;
}
```

## Configuración de los endpoints

```
const url = 'https://xxx.eu-west-1.amazonaws.com/pro/';
const provisionEndpoint = "vdi-provision";
const checkEndpoint = "vdi-check";
```

## API Workspaces

Para la interconexión con los servicios de AWS Workspaces se ha utilizado la API que ofrece AWS. Se puede consultar la documentación en :

<https://docs.aws.amazon.com/workspaces/latest/api/welcome.html>

## Funciones Lambda

Tal como se ha visto antes, la parte de frontal delega en dos funciones Lambda en AWS el aprovisionamiento de la máquina VDI y la comprobación del estado de la creación de la máquina, ya que ésta tarda varios minutos en crearse.

## Función de provisión

La función de provisión se encarga de llamar a la API de Workspaces para pedir la creación de la máquina para el usuario que solicita la creación. Hay una serie de parámetros que tenemos en forma de variables, que deben informarse para la instalación concreta:

1. **ComputeTypeName**: en nuestro caso pedimos STANDARD
2. **RunningMode**: en nuestro caso ponemos AUTO\_STOP
3. **bundle\_id**: el ID del bundle
4. **directory\_id**: el ID del directory

La función lambda se puede ver a continuación (sólo partes relevantes). Como puede verse se están provisionando máquinas de 80G de root FS, que se paran a los 60 minutos de inactividad, y con 10G de disco. El tamaño de la máquina se define en ComputeTypeName.

```
const AWS = require("aws-sdk");
const workspaces = new AWS.WorkSpaces({apiVersion: '2015-04-08'});

let params = {
  Workspaces: [
    {
      BundleId: process.env.bundle_id,
      DirectoryId: process.env.directory_id,
      UserName: "",
      Tags: [
        // poner tags en caso necesario
      ],
      WorkspaceProperties: {
        ComputeTypeName: process.env.ComputeTypeName || 'VALUE',
        RootVolumeSizeGib: 80, // cambiar en caso necesario
        RunningMode: process.env.RunningMode || 'AUTO_STOP',
        RunningModeAutoStopTimeoutInMinutes: 60, // cambiar en caso
necesario
        UserVolumeSizeGib: 10 // cambiar en caso necesario
      }
    }
  ]
};

params.Workspaces.forEach(function(v){
  if(v.WorkspaceProperties.RunningMode==="ALWAYS_ON"){
    delete v.WorkspaceProperties.RunningModeAutoStopTimeoutInMinutes;
  }
});

module.exports.handler = (event, context, callback) => {
```

```
const done = (err, res) => callback(null, {
  statusCode: err ? '400' : '200',
  body: err ? err.message : JSON.stringify(res),
  headers: {
    'Content-Type': 'application/json',
    'Access-Control-Allow-Origin' : '*',
    'Access-Control-Allow-Credentials' : true
  }
});

let user = null;
if(event.requestContext && event.requestContext.authorizer &&
event.requestContext.authorizer.user){
  user = JSON.parse(event.requestContext.authorizer.user);
}

if(!user){
  done("{message:'unauthorized'}");
  return;
}

let setup = JSON.parse(JSON.stringify(params));
setup.Workspaces[0].UserName = user;
setup.Workspaces[0].Tags.push({"Key":"uid","Value":user});

workspaces.createWorkspaces(setup, function(err, data) {
  if(err!==null){done(err, err.message);}
  else{done(null, data);}
});

};
```

## Función de comprobación

Por otra parte tenemos la función de comprobación del estado de la máquina que se ha pedido provisionar para el usuario dado. La función devuelve tres posibles estados: TERMINATING, STOPPED, PENDING o en caso de que el usuario no tenga permisos, UNAUTHORIZED. La función espera que haya una variable "directori\_id" creada.

```
const AWS = require("aws-sdk");
const workspaces = new AWS.WorkSpaces({apiVersion: '2015-04-08'});
```

```
module.exports.handler = (event, context, callback) => {

  const done = (err, res) => callback(null, {
    statusCode: err ? '400' : '200',
    body: err ? err.message : JSON.stringify(res),
    headers: {
      'Content-Type': 'application/json',
      'Access-Control-Allow-Origin' : '*',
      'Access-Control-Allow-Credentials' : true
    }
  });

  let user = null;
  if(event.requestContext && event.requestContext.authorizer &&
event.requestContext.authorizer.user){
    user = JSON.parse(event.requestContext.authorizer.user);
  }

  if(!user){
    done("{\"message:'unauthorized'\"}");
    return;
  }

  let setup = {
    DirectoryId: process.env.directory_id,
    UserName: user
  };

  workspaces.describeWorkspaces(setup, function(err, data) {
    if (err){
      console.log('error: ' + err.message);
      done(err, err.message);
    }else{done(null, data);}
  });

};
```

## Repositorio Github

En el siguiente repositorio podréis encontrar el código que habilita la funcionalidad comentada:

<https://github.com/UOC/aws-workspaces-provisioning>

En breve, la creación del API GW que da acceso a las funciones Lambda, así como las funciones en sí, se podrán crear con el [framework serverless](https://serverless.com) (serverless.com)

## VPN y redes

La conexión con nuestra infraestructura la hemos realizado con el servicio VPN Site-to-site de AWS. Este servicio establece dos túneles en modo activo-pasivo que redirige de uno a otro en caso de incidencia en pocos segundos y nos permite acceder a y desde la infraestructura on-premise al servicio de VDI en AWS.

### Creación de servicio de VPC en AWS

1. Acordar un rango privado que nuestra red privada admita y reconozca como LAN privada.
2. Creamos la infraestructura de red en AWS:
  - a. Creamos una VPC con el rango definido en el punto 1.
  - b. Generamos 4 subnets en AWS a partir de la red del punto 1.
    - i. 2 públicas en diferentes zonas de disponibilidad (eu-west-1 y eu-west-2 en nuestro caso) que crearán IPs públicas-privadas en cada asignación de IP.
    - ii. 2 privadas en diferentes zonas de disponibilidad (eu-west-1 y eu-west-2 en nuestro caso)
  - c. Creamos un set de DHCP OPTIONS para que los equipos que se añadan a estas subnets privadas tengan la configuración de DNS interna de UOC y el NTP de UOC.
  - d. Creamos un NAT Gateway en AWS y lo asociamos a las Subnets públicas.
  - e. Creamos un Internet Gateway para dar salida a internet y los vinculamos a las subnets.
  - f. Asignamos rutas estáticas a las subnets para dotarlas de salida a internet:
    - i. 0.0.0.0/0 → INTERNET GATEWAYEn nuestro caso, todo lo hemos provisionado utilizando Infraestructura como código (IaC) utilizando el lenguaje de Hashicorp: TERRAFORM.

### Creación del servicio VPN Site-to-Site

Como hemos comentado, establecemos 2 túneles con nuestra infraestructura para dotar de alta disponibilidad en las comunicaciones UOC ↔ AWS.

Utilizamos el servicio Site-to-site VPN de AWS que se compone de 3 elementos:

1. Customer Gateway
2. VPN Gateway
3. VPN Connection (Site-to-Site)

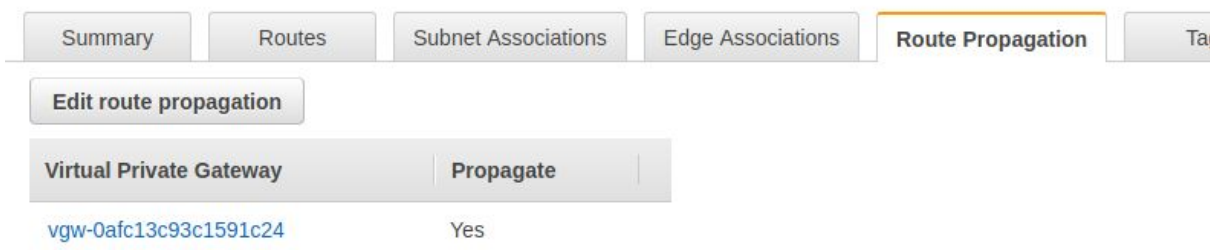
Al generar la infraestructura, nos pone a disposición dos “terminaciones” que serán los 2 túneles que tendremos que establecer con nuestra electrónica de red.

Una vez el túnel (los dos) esté establecido, configuramos el resto de elementos para “encaminar” las peticiones internas de forma correcta hacia y desde AWS utilizando las redes internas que definimos en el punto 1 del apartado anterior.

Seguidamente, establecemos las rutas a las IPS INTERNAS de UOC que los servicios en AWS deben encaminar hacia la VPN. El resto, por costes y velocidad, debe encaminarse hacia el Internet Gateway:

1. Añadimos las rutas en la VPN Connection
2. En las tablas de rutas de las subnets (privadas o públicas o ambas), establecemos que se propaguen las rutas del Virtual Private Gateway:

Route Table: rtb-087201918933fa849



The screenshot shows the AWS console interface for a Route Table. The 'Route Propagation' tab is selected, and the 'Propagate' checkbox is checked for the associated Virtual Private Gateway.

Virtual Private Gateway	Propagate
vgw-0afc13c93c1591c24	Yes

Y podemos verificar que en la pestaña “ROUTES” están efectivamente las rutas que deben ir hacia la VPN:

0.0.0.0/0 → Internet Gateway

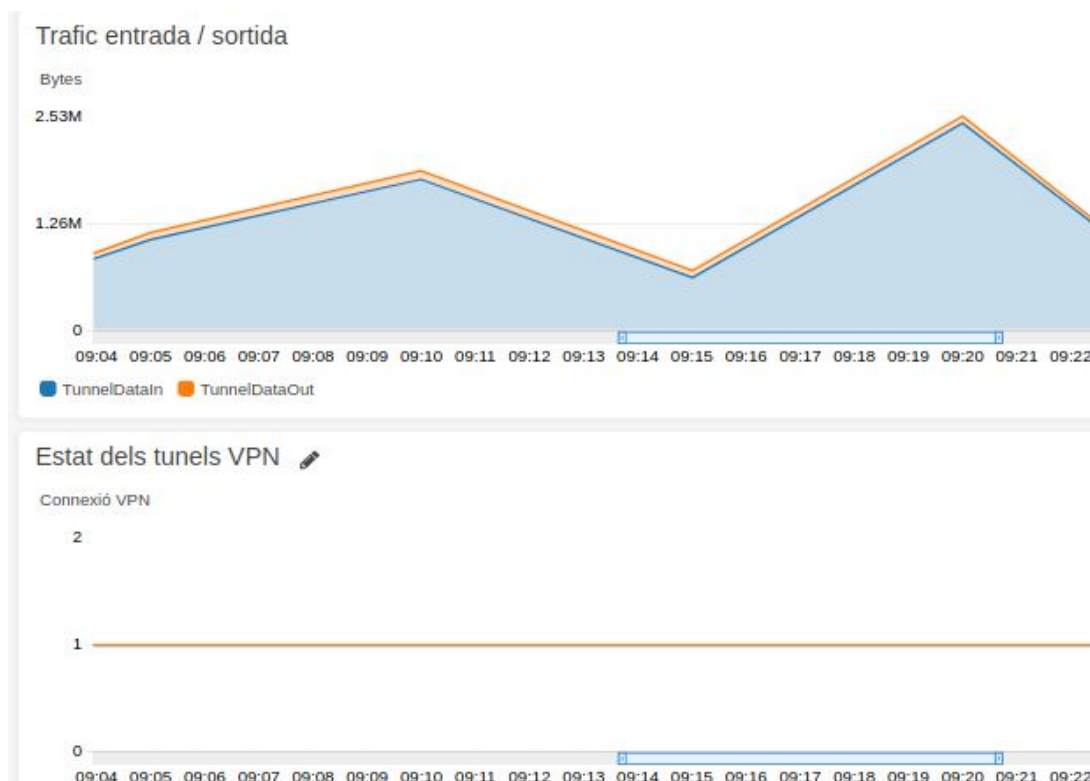
x.x.x.x/x (RANGO PRIVADO DE UOC) → Virtual Private Gateway.

x.x.x.x/x (RANGO 2 PRIVADO DE UOC) → Virtual Private Gateway.

x.x.x.x/x (RANGO N PRIVADO DE UOC) → Virtual Private Gateway.

Adicionalmente, generamos 2 gráficos en CloudWatch que monitorizan el estado de los túneles y la cantidad de tráfico (entrada-salida) que circula:





En nuestro caso, todo los hemos **provisionado utilizando Infraestructura como código (IaC)** utilizando el lenguaje de Hashicorp: TERRAFORM.

## Conexión con AD

Para autenticar a los usuarios en el propio Workspace, Amazon hace uso de AD (Microsoft Active Directory), esta autenticación identificará al usuario y le permitirá acceder a su Workspace (VDI) que previamente tendrá creado mediante el anteriormente explicado sistema de provisión.

El AD que se utilizará puede ser de 3 tipos:

1. **AD gestionado por AWS:** un AD en el cloud en el que deberán darse de alta los usuarios y gestionarlo de la misma manera que un AD on-premise. Éste sirve además de para Workspace para cualquier otro servicio que lo necesite (RDS para bases de datos SQL Server, aplicaciones .NET, etc.).
2. **Simple AD:** Es una instancia de un servidor Linux corriendo un servidor SAMBA (compatible con AD).
3. **AD Connector:** Es un proxy a través del cual se conectará al AD on-premise a través de (en nuestro caso) una VPN o si se dispone del Direct Connect.

En nuestro caso hemos utilizado el AD Connector para acceder al AD que ya disponíamos.

La configuración de este AD Connector es bastante sencilla y es la opción más recomendada si ya disponíamos de un AD. Se requiere configurar los siguientes datos en el asistente de creación del directorio:

1. **Tamaño del directorio:** pequeño (máximo 5000 usuarios y 150 autenticaciones por minuto) o grande (hasta 75000 usuarios y 2500 autenticaciones por minuto).
2. **VPC y subredes:** Se debe indicar la VPC donde se encuentran las subredes en las que se creará el conector. Debe ser accesible o ser la misma en donde se crearán los VDI y tener acceso (por VPN o DirectConnect) al servidor AD. Se recomienda crear una VPC y al menos dos subredes para contener tanto el conector AD como los escritorios virtuales.
3. A continuación se piden datos de acceso al AD:
  - **Nombre de la organización:** debe ser un nombre único a nivel global, no debe existir otro ni siquiera de otro cliente ajenos a nosotros. Se utilizará para crear las urls a servicios vinculados a nuestros workspaces. Ejemplo: uoc-ad
  - **Nombre del DNS del directorio:** indicaremos el dominio completo de nuestro AD, por ejemplo: interna.uoc.edu
  - **Nombre NetBIOS del directorio:** es opcional pero si se dispone de más de uno es conveniente indicar el adecuado. En nuestro caso por ejemplo sería: INTERNA
  - **Direcciones IP de DNS:** se deberán especificar dos DNS de dos de nuestros servidores AD, no se trata de un DNS para resolución de nombres de internet sino de un DNS Dinámico donde se registrarán las máquinas en nuestro dominio.
  - **Nombre de usuario de la cuenta de servicio:** se trata de un usuario/password de un usuario administrador o con permisos para administrar el registro de máquinas en nuestro AD. Se recomienda crear un usuario específico y no utilizar un usuario nominal ya que si el usuario por ejemplo se cambiase el password provocaría que dejase de funcionar el AD Connector.

Tras introducir estos datos y pasados varios minutos tendremos disponible el AD Connector, pero aún será necesario realizar algunos ajustes.

Como el conector es nuevo posiblemente no se encuentre registrado, al crear un workspace se registrará pero lo podemos hacer manualmente seleccionando el directorio y en el desplegable "Actions" seleccionamos "Register". Se nos solicitará en este paso que indiquemos las subredes donde se crearán los VDI, tras esto realizará el registro.

A continuación seleccionamos el directorio ya creado y registrado y en el desplegable "Actions" elegimos la opción "Update Details". Aquí tenemos bastantes opciones que podemos configurar, pero hay 4 que son importantes y que se detallan a continuación:

1. **Target Domain and Organizational Unit:** Es donde queremos que se registren las máquinas VDI, podemos hacer uso del buscador que se incorpora en esta opción para encontrar más fácilmente la rama que queremos seleccionar. Si no se configura esta opción los VDI se registrarán en la rama por defecto (OU=Computers, DC= (...)). El tener una rama para estos VDI nos facilita la administración y el mantenimiento con (por ejemplo) System Center sin afectar a otras máquinas pertenecientes al dominio con necesidades distintas.
2. **Security Group:** Aquí seleccionaremos un SG para controlar la conectividad (como haría un firewall), es importante crear uno sobretodo si las VDI las creamos en una subred pública donde se encontrarán totalmente expuestas en internet. El consejo es permitir todo el tráfico entre los miembros del propio SG, permitir todo entre nuestras redes internas (las conectadas mediante VPN o Direct Connect) y denegar por defecto el resto para evitar conexiones desde internet a los VDI.
3. **Access to Internet:** Nos aseguraremos que se encuentra esta opción en Enable para permitir que los VDI puedan acceder a internet. Si se encuentra en Disable y al intentar activarlo no nos lo permite significa que tenemos mal configurada la VPC (rutas, Internet Gateway o NAT Gateway, etc.). Tras corregir el problema nos permitirá activar esta opción.
4. **Access Control Options:** Permite seleccionar cómo se permite acceder a los VDI, a parte de si queremos utilizar certificados digitales o no, lo importante es el apartado "Other Platforms" donde deberemos activar las opciones que nos interesen, por ejemplo el acceso desde linux y Web por defecto se encuentran desactivadas, es interesante activar todas las opciones para así dar el máximo de facilidad al usuario final de acuerdo a sus necesidades.

El resto de opciones no son necesarias pero sí recomendable revisarlas, por ejemplo "User Self Service Permissions" permite o no que los usuarios puedan adaptarse los recursos de sus propios workspaces (más CPU, más memoria, más espacio en disco, etc.) con lo que quizás es recomendable restringir estas opciones para no incurrir en gastos innecesarios.

## Security Group de los WORKSPACES

Al crear un nuevo Directory, se generan 2 SECURITY GROUPS nuevos:

1. Para la comunicación del servicio de conexión con el AD privado con el nombre: **DIRECTORYID\_controllers**
2. Donde se añaden los WORKSPACES: **DIRECTORYID\_workspacesMembers**

Por defecto, el SG se configura de la siguiente forma:

- **DIRECTORYID\_controllers:** Se autoconfigura de manera que permite el tráfico de entrada-salida total (es importante que el DIRECTORY se ejecute en las subnets privadas).
- **DIRECTORYID\_workspacesMembers:** Por defecto, se configura de manera que permite el tráfico saliente (hacia Internet o hacia nuestras redes privadas de VPN) y con todo denegado de entrada.  
Esto hace que no tengamos conexión en el sentido UOC → AWS, por lo tanto aquí es necesario abrir las redes privadas (si no queremos exponer todos los puertos de todos los VDIs a INTERNET) de UOC con los servicios que necesiten conexión con los VDI.

## Imagen plantilla y promoción

### Instalación inicial del VDI “maqueta”

Workspaces proporciona un w10 sobre una plataforma windows server 2016, a partir de ella, una vez se tiene conectividad con el repositorio de software UOC, se procede a instalar todo lo necesario para el punto de trabajo estándar UOC.

Como el equipo se valida en el dominio al entrar en sesión, se pueden aplicar las políticas de grupo en el directorio activo (gpo's) que se deseen en la rama del AD especificada.

Antes de ‘cerrar’ la maqueta se puede personalizar el aspecto para que tenga la imagen de marca que se desea o con los accesos directos en el escritorio que se precisen.

Después se podrán hacer las modificaciones que se precisen (replicando de la maqueta), poniendo en mantenimiento los equipos ya creados.

### Promoción de la maqueta

Una vez dispongamos del software necesario para el punto de trabajo instalado en un VDI, haremos una “IMAGEN” a partir de esa. Esta imagen “congela” el estado (software) de esa VDI y nos permite hacer una plantilla. Con esta imagen base crearemos un BUNDLE donde aplicaremos la “talla” (capacidad del HW Virtual) de cada escritorio virtual que se ejecute.

En cada nueva provisión especificamos parámetros (ver apartado de [ejecución](#) más arriba) donde uno de ellos es el “BUNDLE ID” donde le decimos *QUÉ* BUNDLE (image+parámetros) arranca en el momento de la primera provisión de VDI. Este sistema nos permite tener una plantilla con el SW instalado controlado.

## Restauración de un VDI

Una vez el VDI esté creado y promocionado, los cambios son persistentes en cada VDI, no obstante, nos dejamos la posibilidad de hacer un “REBUILD” de cada(s) VDI por si hay que volver al estado inicial debido a una incidencia o mal funcionamiento utilizando el BUNDLE original.