THIS IS A TEMPLATE

Implementing Cloudflare for SaaS

Migrating custom hostnames to Cloudflare

Generally, we will be using this public guide and the standard Cloudflare for SaaS configuration as a reference in order to configure Cloudflare for SaaS.

This is a high-level migration plan intended for orientation purposes.

Disclaimer

These are general guidelines that may be useful depending on various factors, such as different use cases. It is important to note that customers are responsible for understanding the impact of their actions. These recommendations should be adapted to fit each customer's specific requirements and needs.

For any questions or issues, please contact your account team or support.

TABLE OF CONTENTS

Setup

SaaS Provider Setup

The following information (variables) below are used for SaaS-Company-Name-Here's Cloudflare for SaaS setup.

SaaS Provider: SaaS-Company-Name-Here

Cloudflare Zone: saas.provider (*replace with your Zone/Domain here*)

Cloudflare Zone ID: abcdefg0987654321 (*replace with your ZoneID here*)

Fallback Origin: fallback.saas.provider

Custom Hostname example: saas-provider.customer1.com (*replace with your end-customer's custom hostname*)

Migration Steps

1. Create fallback origin

- Create a proxied A, AAAA, or CNAME record fallback.saas.provider pointing to the IP address of your fallback origin (where Cloudflare will send custom hostname traffic by default).
- Designate that record as your fallback origin.
- Once the SaaS provider has added the fallback origin, confirm that its status is Active.

2. Create a proxied CNAME Target

This CNAME target will be used by your end-customers later.

Using a CNAMEd record in front of your origin allows the SaaS provider to update the fallback address without asking your customer to update their DNS, and using a customer specific record for each domain allows the SaaS provider to manage the routing of that customer's traffic without their intervention should the need arise in the future.

```
Unset
*.customers.saas.provider   CNAME   fallback.saas.provider
```

Specifically for the use case of regionalization (Regional Services, part of the Data Localization Suite), please note the behavior with Cloudflare for SaaS. The SaaS provider might want to create a regionalized CNAME Target for end-customers with regionalization requirements.

```
Unset
eu-customer.saas.provider CNAME fallback.saas.provider  REGION European Union
us-customer.saas.provider CNAME fallback.saas.provider  REGION United States
```

In this example, this CNAME Target will be used as a target by the end-customer's custom hostname in their authoritative DNS. With this, the custom hostname will be regionalized in the European Union.

3. Coordinate with end-customers

Before continuing, SaaS-Company-Name-Here communicates and clarifies with their end-customers the plan for:

- Certificate validation; and
- Hostname validation.
- (Optional) in case end-customers are also Cloudflare customers and proxying traffic through Cloudflare, **Orange-to-Orange (O2O)** will be required. Please review the product compatibility. Generally, it's recommended for end-customers to DNS Only / gray-cloud the hostnames used by the SaaS provider.

If **minimizing downtime** is an important requirement, then it is generally recommended to go with hostname pre-validation methods and certificate Delegated Domain Control Validation (DCV).

Please note that for **wildcard custom hostnames**, end-customers must add an extra CNAME record (Delegated DCV) for every wildcard custom hostname they have. Without this, end-customers will have to add the TXT record validation every 60 days. Wildcards also behave differently, as noted in the documentation.

One hostname pre-validation method is TXT validation, when the end-customer adds a TXT record to their authoritative DNS to verify domain ownership.
Alternatively, end-customers that do not wish to add DNS records to their authoritative DNS can perform HTTP validation by adding HTTP tokens to their origin web server.

For each custom hostname, Cloudflare issues two certificates bundled in chains that maximize browser compatibility (unless the SaaS provider uploads custom certificates).

4. Create custom hostname

Below is an example of a POST API call to create the custom hostname example saas-provider.customer1.com with specific SSL settings, such as minimum TLS Version and TLS 1.3, as well as (optional) custom metadata (review the *Taking advantage of Custom Metadata* section for more details).

Unset

```
curl --request POST \
 --url
https://api.cloudflare.com/client/v4/zones/abcdefg0987654321/custom_hostnames \
 --header 'Content-Type: application/json' \
 --header 'X-Auth-Email: <USER_EMAIL>' \
 --header 'Authorization: Bearer <API_TOKEN>' \
 --data '{
```

```
 "custom_metadata": {
    # Here you can add custom metadata.
    "customer_id": "12345",
    "redirect_to_https": true,
 },
 "hostname": "saas-provider.customer1.com",
 "ssl": {
  "bundle_method": "ubiquitous",
  "certificate_authority": "google",
  "method": "txt",
  "settings": {
    # Here you can add SSL settings. Review the API documentation for details.
    "http2": "on",
    "tls_1_3": "on"
  },
  "type": "dv",
  "wildcard": false
 }
}'
```

Important is the output of this API call, as SaaS-Company-Name-Here will have to share the *ownership_verification* or *ownership_verification_http* information with their specific end-customer.

Below is an example of a TXT validation output. The end-customer will have to add a TXT record with that *name* and *value* at their authoritative DNS provider.

```
Unset
{
"result": [
  {
  "id": "3537a672-e4d8-4d89-aab9-26cb622918a1",
  "hostname": "saas-provider.customer1.com",
  // ...
  "status": "pending",
  "verification_errors": ["custom hostname does not CNAME to this zone."],
  "ownership_verification": {
    "type": "txt",
    "name": "_cf-custom-hostname.saas-provider.customer1.com",
    "value": "0e2d5a7f-1548-4f27-8c05-b577cb14f4ec"
  },
  "created_at": "2022-03-04T19:04:02.705068Z"
  }
]
}
```

Once the custom hostname is activated, the end-customer can remove the TXT record.

4.1 Configure a Custom Origin for a Custom Hostname

A custom origin server lets the SaaS provider send traffic from one or more custom hostnames to somewhere besides their default fallback origin.

Note that **custom origins update the SNI value** for the requests downstream to match the custom origin host. Review the connection request details and SNI rewrites for more information.

Please note that all SNI rewrite usage is subject to the Service-Specific Terms (ToS).

To use a custom origin, select that option when creating a new custom hostname in the dashboard or include the *"custom_origin_server": your_custom_origin_server* parameter when using the POST API call.

4.2 Configure the Cloudflare Tunnels and expose web applications

A SaaS-Company-Name-Here requirement is to expose their web applications via Cloudflare Tunnels public hostnames or private networks (private IPs). It is generally **recommended to use the private networks** method, as it allows for better scalability due to preserving the Host header from the client.

Host header configuration might not be required though, depending on how your servers are configured as well, or if you have an ingress Tunnel catch-all configuration in place.

First, the *cloudflared* will have to be installed on the different origin servers. Then expose the public hostname or IP/CIDR range.

For private networks, please ensure to connect them to a virtual network (vnet).
For public hostnames, please ensure that there is an active Edge Certificate in your Zone beforehand.

Note that using the Cloudflare Tunnel to expose web applications is completely **optional**. The Load Balancer can also be used directly with the origin IP addresses, and this would potentially simplify the entire setup.

Alternative options to protect origin servers are listed in the developer documentation.

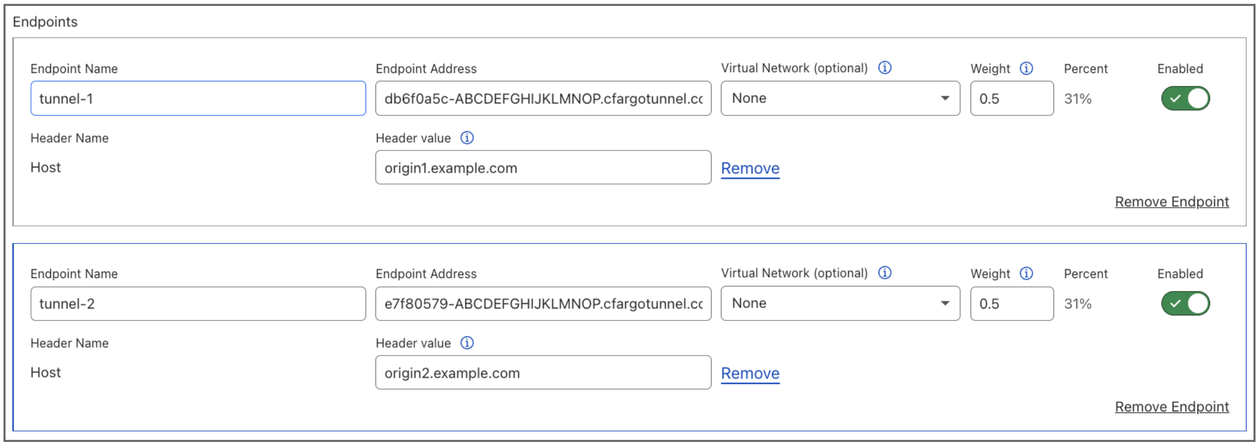
Please note the Cloudflare Tunnel limitations.

4.3 Configure the Load Balancer with Health Monitors

A SaaS-Company-Name-Here requirement is for some custom hostnames to require a **Load Balancer as a custom origin** with session affinity (sticky session). This requires the prior configuration of the load balancer and then adding it as a custom origin server to the specific custom hostname.

The Load Balancer configuration could look like the following: one Origin Pool with origin-1, the Origin Address being *<UUID-1>.cfargotunnel.com*, and origin-2, the Origin Address being *<UUID-2>.cfargotunnel.com*. Additionally, adding the Header value of the public hostnames configured and exposed by each Cloudflare Tunnel.

As an example:



Endpoints

Endpoint Name	Endpoint Address	Virtual Network (optional) ⓘ	Weight ⓘ	Percent	Enabled
tunnel-1	db6f0a5c-ABCDEFGHIJKLMNOP.cfargotunnel.cc	None	0.5	31%	✓

Header Name	Header value ⓘ	
Host	origin1.example.com	Remove

Remove Endpoint

Endpoint Name	Endpoint Address	Virtual Network (optional) ⓘ	Weight ⓘ	Percent	Enabled
tunnel-2	e7f80579-ABCDEFGHIJKLMNOP.cfargotunnel.cc	None	0.5	31%	✓

Header Name	Header value ⓘ	
Host	origin2.example.com	Remove

Remove Endpoint

Alternatively, if using private networks exposed through the Cloudflare Tunnel instead, use the private IPs as Origin Address within the Origin Pool and selecting the associated virtual network.

For the Health Monitors, simply configure the type HTTP/S with HEAD method, expecting a 200 status code and following redirects.

With this, the Load Balancer should indicate the origin servers to be healthy and reachable, if the Tunnels are active and healthy. Please review the local connection preference.

5. The end-customer validates hostname and certificate

During the custom hostname creation, SaaS-Company-Name-Here – the SaaS provider – shares with the end-customer the necessary information, in order to (pre-)validate hostname and certificate.

In this example below, the end-customer uses TXT validation for the hostname by creating a TXT record at their authoritative DNS. This only needs to be done once to validate the hostname, then the end-customer can remove the TXT record.

Unset

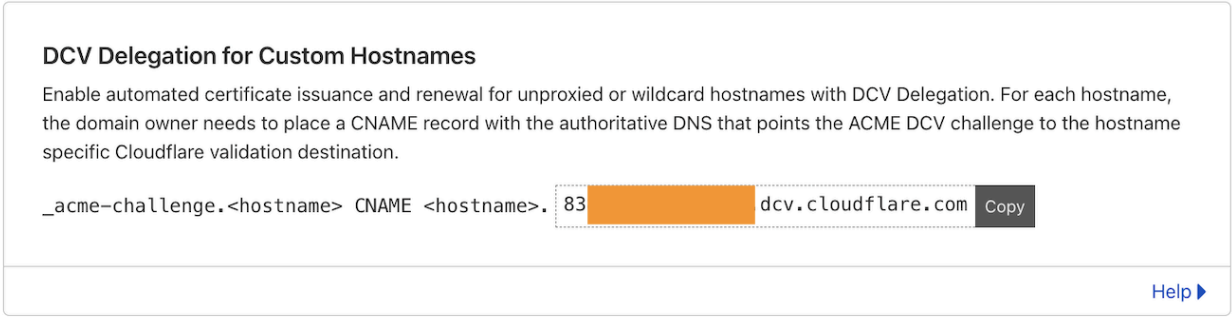
```
_cf-custom-hostname.saas-provider.customer1.com   TXT
0e2d5a7f-1548-4f27-8c05-b577cb14f4ec
```

In this example below, the end-customer uses Delegated DCV for the certificate's validation and future auto-renewal by creating a CNAME record at their authoritative DNS.

Unset

```
_acme-challenge.saas-provider.customer1.com  CNAME
saas-provider.customer1.com.<COPIED_HOSTNAME>.
```

When referring to *"<COPIED_HOSTNAME>"*, we are talking about the one copied from the Cloudflare Dashboard in the section *DCV Delegation for Custom Hostnames*. See screenshot:



DCV Delegation for Custom Hostnames

Enable automated certificate issuance and renewal for unproxied or wildcard hostnames with DCV Delegation. For each hostname, the domain owner needs to place a CNAME record with the authoritative DNS that points the ACME DCV challenge to the hostname specific Cloudflare validation destination.

```
_acme-challenge.<hostname> CNAME <hostname>. 83[redacted]dcv.cloudflare.com
```

6. The end-customer creates a CNAME record

With the hostname and certificate now validated, the end-customer can finish the custom hostname setup by creating a CNAME record at their authoritative DNS that points to SaaS-Company-Name-Here’s CNAME target (created in step 2).

Unset

```
saas-provider.customer1.com   CNAME   customer1.customers.saas.provider
```

This will finally proxy the custom hostname to the fallback origin.

Repeat these 6 steps for every custom hostname. This process can be automated via API calls, scripting or Terraform. Review the section on Automation.

6.1 Apex proxying requires Static IPs or BYOIP

With apex proxying, end-customers need to create an A record for their hostname that points to the IP prefix allocated to the SaaS provider’s account.

If your end-customer uses an A record at their authoritative DNS provider, they need to point their hostname to the IP prefixed allocated for your account.

Unset

```
customer2.com.  60  IN  A   192.0.2.1
```

Please note that Static IPs or BYOIP is required for the IP validation.

Please also review the hostname priority (Cloudflare for SaaS) documentation.

Taking advantage of Custom Metadata

Custom metadata can be used in a Rule Expression, such as i.e. in WAF Custom Rules or Configuration Rules.

Taking the example of section 4 (Create custom hostname), the SaaS provider can use the Configuration Rules to identify the custom metadata and apply settings based on the value.

Unset

```
lookup_json_string(cf.hostname.metadata, "redirect_to_https") eq "true"
```

In this example above, SaaS-Company-Name-Here can configure Configuration Rules with this expression, applying different settings, such as Automatic HTTPS Rewrites, to matching custom hostnames.

The usage of custom metadata is entirely optional and some limitations may apply.