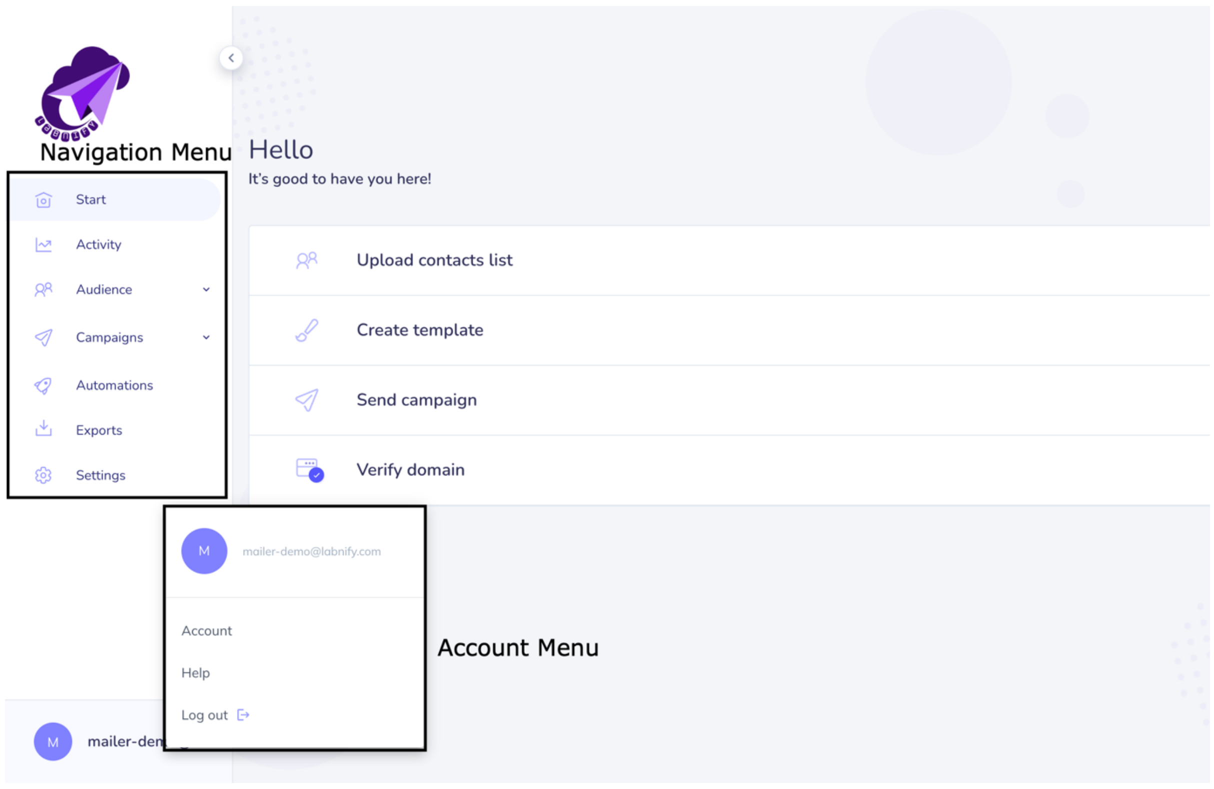
📘 LABNIFY USER MANUAL

Managing your account

This tutorial is meant to help first-time users navigate their account. When you log in for the first time you will see your "Start" screen. This screen offers a set of quick navigation points that take you to the most relevant screens for first-time set up. Other than that you can find the navigation menu and the account menu here.



Navigation menu
Access to this menu will always be available on your dashboard, regardless of which screen are you looking at currently.

Start - This screen brings you to back to the Start screen.

Activity - This screen holds the activity graph showing your recent sending trends as well as detailed email logs of each of your campaigns.

Audience - The contact management screen. Here you can upload your contacts and sort them between lists and dynamic segments.

Campaigns - The Labnify Email Magic Campaign Creator - our powerful tool for creating marketing newsletters. You can find your past campaigns list and their statistics here too.

Automation – Here you can create automation templates for your campaigns.
Exports – On this screen, you can manage all your exports

Settings – Here you can change all your necessary settings like API keys, Domains, SMTP credentials etc.

Start menu

This menu offers quick links to the most relevant parts of your dashboard, they can also be found under the navigation menu and account menu, depending on your preference.

Upload Contacts List - Part of the "Contacts" screen that takes you directly to the contact uploading tool.

Create Template - Part of the "Templates" screen that takes you directly to the template editor.

Send Campaign - Part of the "Campaigns" screen that takes you directly to the campaign creation tool.

Verify Domain - A very useful link that takes you directly to the domain verification screen. Domain verification is an important step to fully activate your account and lift any possible sending limits.

Account menu

If you click your account name in the bottom-left corner of your dashboard, a pop-up menu with your profile and account settings will appear.

Billing - Change plan and billing details and view payment history, usage and receipts.

Profile - Update your profile information that can be used in your mailings.

Security – Manage your Password, 2FA and other security settings for your account.

Help – Go to our live chat to solve all your basic queries in real time.

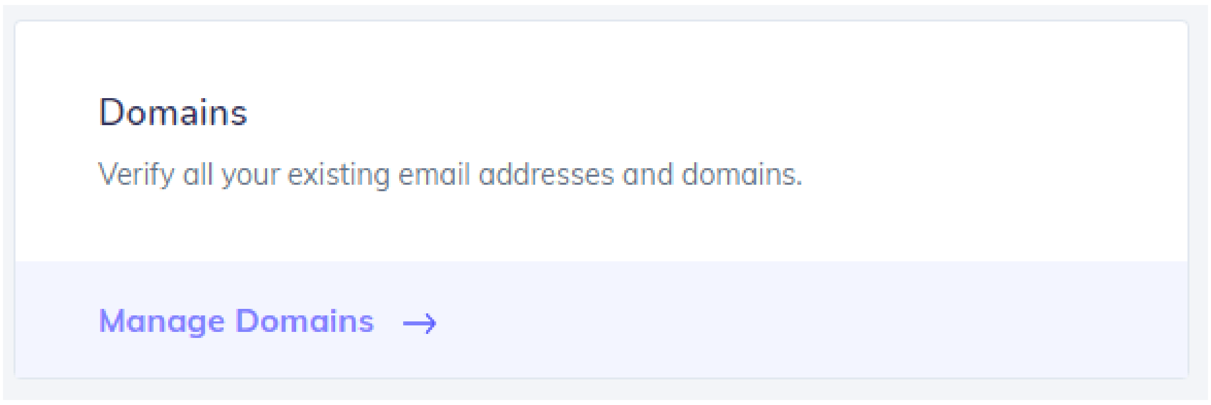
Log out – Safely log out of your Labnify account.

Verifying your Domain

Completely whitelabel your emails using your own domain to improve email deliverability.

Getting Started

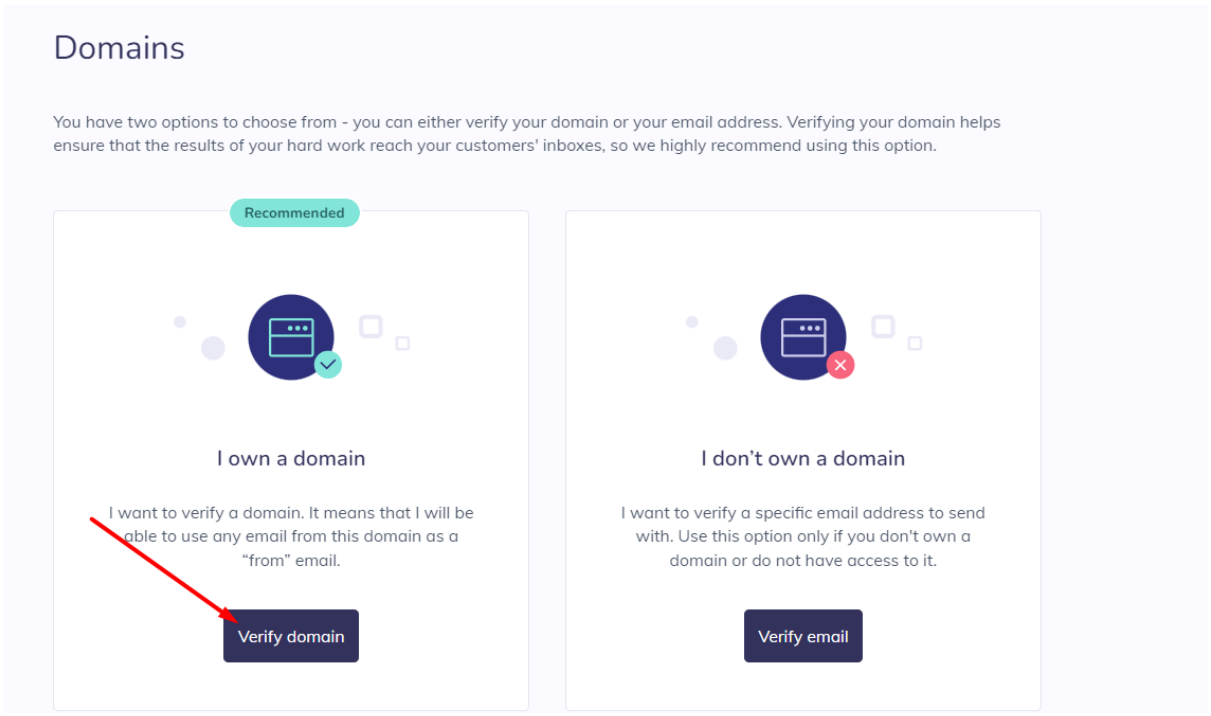
In order to verify your domain, navigate to **Settings>Domains>Manage Domains** screen:



Domains

Verify all your existing email addresses and domains.

Next, click "Start Verification" in order to bring up the "Domains" modal. In here you can choose between domain and email verification. Choose the domain option to start the verification process.



Domains

You have two options to choose from - you can either verify your domain or your email address. Verifying your domain helps ensure that the results of your hard work reach your customers' inboxes, so we highly recommend using this option.

Recommended

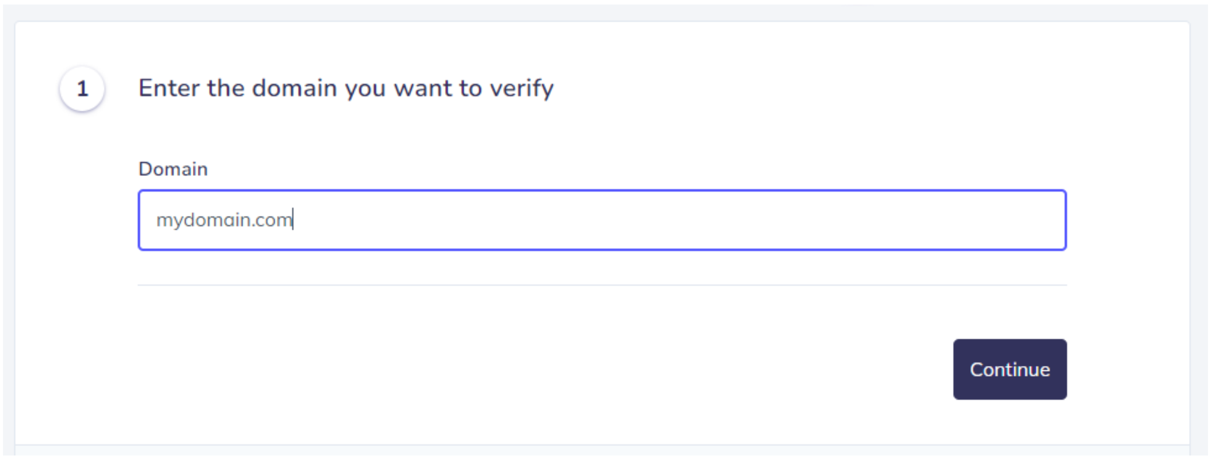
I own a domain

I want to verify a domain. It means that I will be able to use any email from this domain as a "from" email.

I don't own a domain

I want to verify a specific email address to send with. Use this option only if you don't own a domain or do not have access to it.

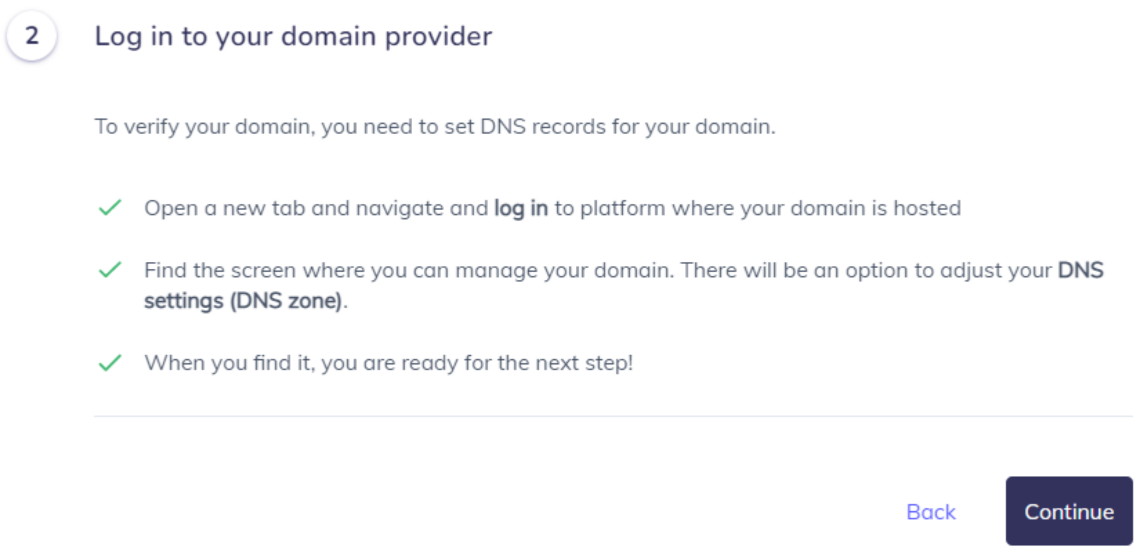
1) Input your domain name and hit "Continue". The domain must be owned by you and not registered with any other account


1 Enter the domain you want to verify

Domain

mydomain.com

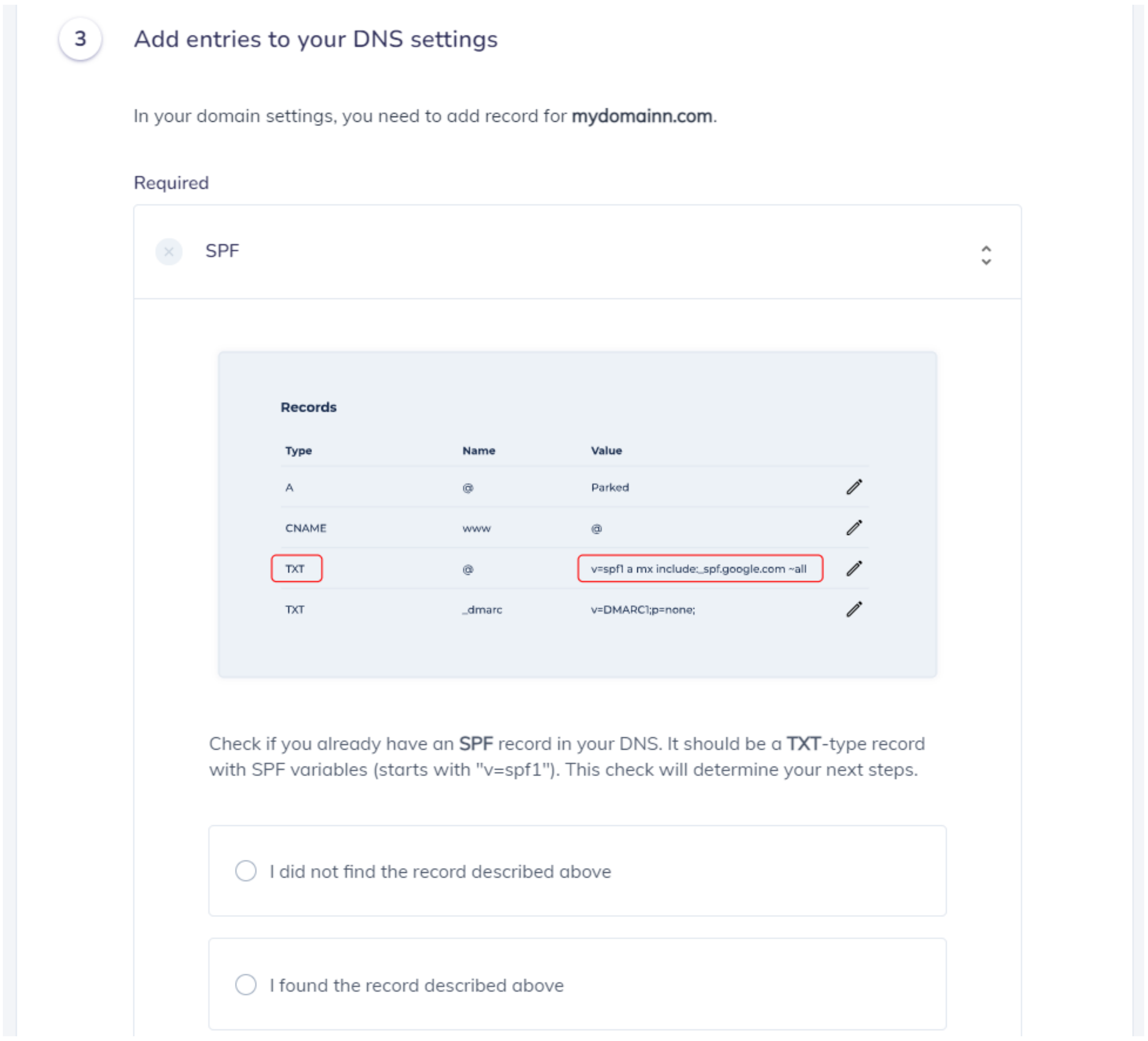
2) Log in to your domain provider


2 Log in to your domain provider

To verify your domain, you need to set DNS records for your domain.

- ✓ Open a new tab and navigate and **log in** to platform where your domain is hosted
- ✓ Find the screen where you can manage your domain. There will be an option to adjust your **DNS settings (DNS zone)**.
- ✓ When you find it, you are ready for the next step!

3) Add entries to your DNS settings


3 Add entries to your DNS settings

In your domain settings, you need to add record for **mydomainn.com**.

Required

SPF

Records

Type	Name	Value
A	@	Parked
CNAME	www	@
TXT	@	v=spf1 a mx include:_spf.google.com ~all
TXT	_dmarc	v=DMARC1;p=none;

Check if you already have an **SPF** record in your DNS. It should be a **TXT**-type record with SPF variables (starts with "v=spf1"). This check will determine your next steps.

○ I did not find the record described above

○ I found the record described above

When properly set up, you can hit the "Verify record" button to continue.

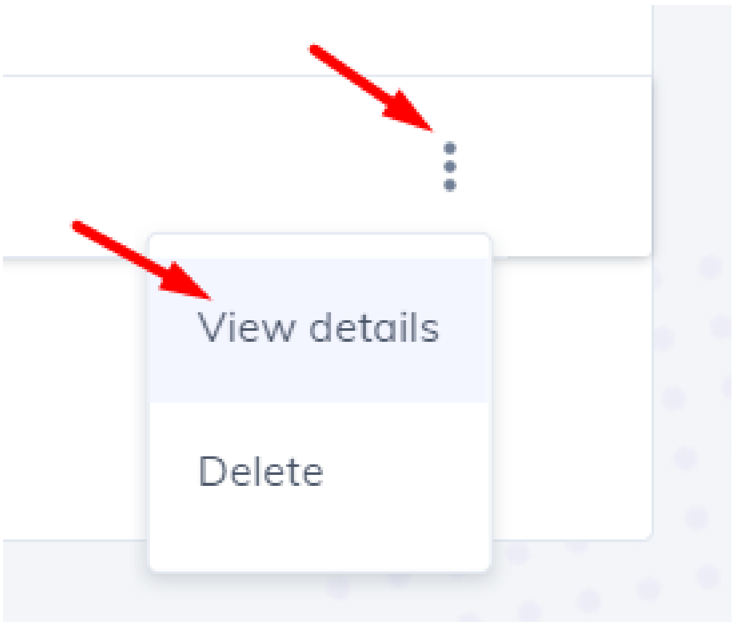
4) Confirmation

If everything was set up correctly, you should be at a confirmation step of the verification process. This informs you that you can now use any email address related to your domain to send emails.

5) Set default sender

When there is a default sender set on the account, emails are still branded using the default sender, even if sent from a non-verified domain (please note: you have to verify your domain before you can be allowed to send emails).

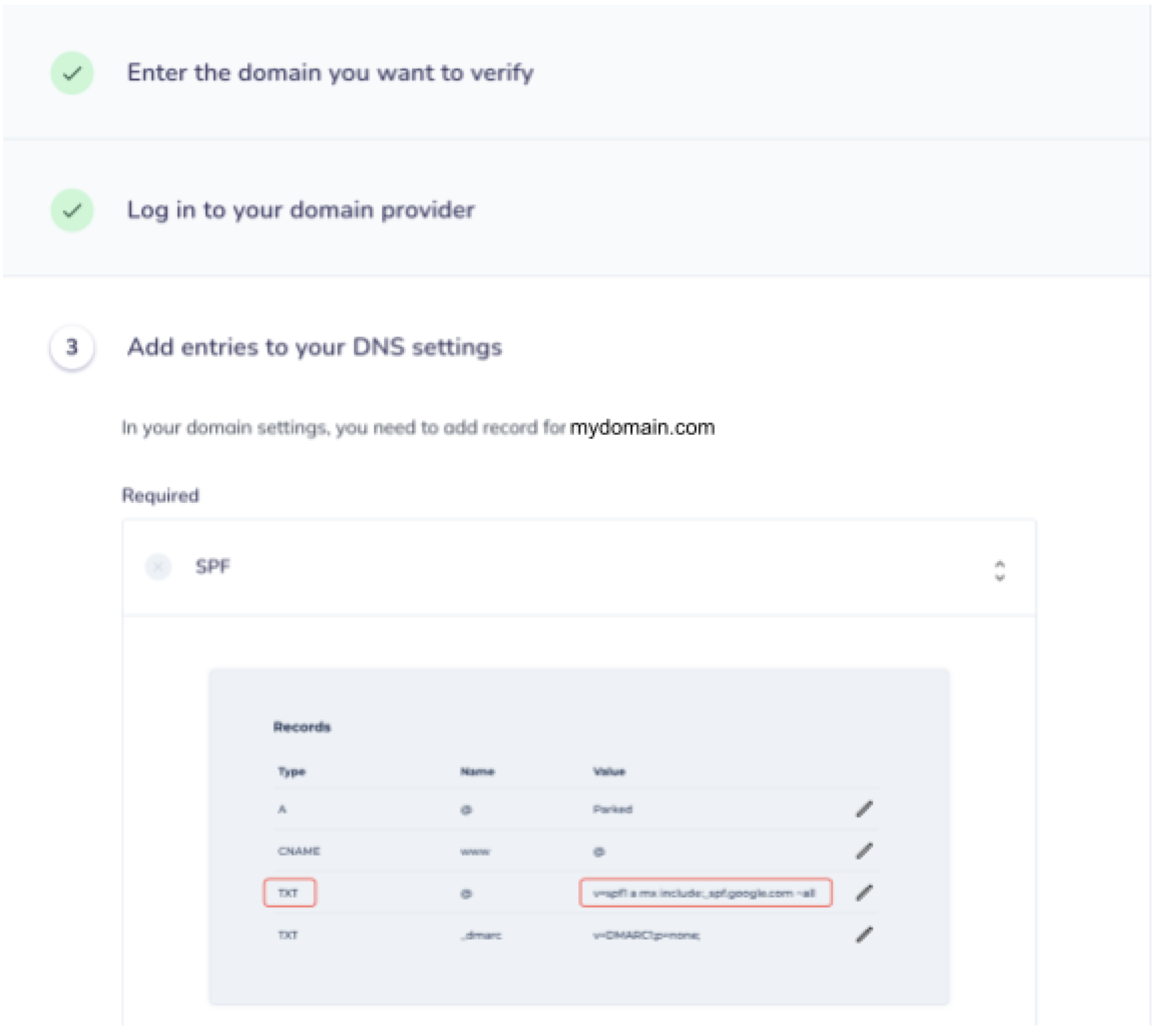
When your domain is added to the account, it's now visible in the domains list. If a blue star is present next to the domain name, it indicates that it's set as a Default Sender. You can enter a domain's details by clicking **View details** next to the domain name.



The following screen shows detailed information about your domain's verification status.

Here, you can set your domain as default sender and check if your **SPF, DKIM, Tracking, MX and DMARC** records are properly verified.

SPF is mandatory if you expect to experience good email deliverability. We also encourage you to verify DKIM. If you click the name of a record (e.g. "SPF") on this screen, a popup with detailed instructions on how to verify it will be displayed for you. Instructions on verification of every record will also be included below.



Enter the domain you want to verify

Log in to your domain provider

3 Add entries to your DNS settings

In your domain settings, you need to add record for mydomain.com

Required

SPF

Records

Type	Name	Value
A	@	Parked
CNAME	www	@
TXT	@	v=spf1 a mx include:_spf.google.com ~all
TXT	_dmarc	v=DMARC1;p=none;

1. SPF

SPF stands for "Sender Policy Framework". An SPF record is in place to identify which mail servers are authorized to send mail for a given domain. It is used to prevent spammers from sending mail with fraudulent From addresses in that domain.
Though many DNS editors allow for the creation of a SPF record, the SPF record must be entered as a TXT record in your domain's DNS settings. Enter:

Host/Name: @ (This means that the record is pointed at your own domain. Some editors will require the "@" symbol, some will require you to enter your own domain, and others will not let you enter anything. Every DNS Editor is different - you may need to contact your hosting provider for information on how to enter this record correctly).
Value: v=spf1 a mx include:relay.labnify.com ~all
SPF tips: Check to see if there are any other SPF records in your domain's DNS. There can only be one SPF record per domain, so if there is an existing record, just add "include:relay.labnify.com" to that record. *Make sure you remove the quotes.*
For example, if your domain already has the record: v=spf1 a mx include:_spf.google.com ~all, then you would just add: include:relay.labnify.com

The final record would look like this: v=spf1 a mx include:relay.labnify.com include:_spf.google.com ~all

2. DKIM
DKIM stands for "DomainKeys Identified Mail". They allow receiving servers to confirm that mail coming from a domain is authorized by the domain's administrators.
Create a TXT record. Enter:
Host/Name: api._domainkey
Value:k=rsa;t=s;p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCbmGbQMzYeMvxwtNQoXN0waGYaciuKx8mtMh5czguT4EZlJXuCt6V+l56mmt3t68FEX5JJ0q4ijG71BGoFRkl87uJi7LrQt1ZZmZCvrEII0YO4mp8sDLXC8g1aUAoi8TJgxq2MJqCaMyj5kAm3Fdy2tzftPCV/lbdiJqmBnWKjtwIDAQAB
TIP: DNS settings can have as many DKIM records as needed.
In some DNS settings, the Host/Name field may require you to enter "api._domainkey.yourdomain.com", replacing your domain with your actual domain.

3. Tracking
Labnify "tracks" opens, clicks, unsubscribes, etc. To do that, we must rewrite links and use web pages. Setting up a "tracking domain" brands these rewritten links and pages with your own domain.
Create a CNAME record. Enter:
Host/Name: tracking
Value: relay.labnify.com

3.1 Tracking Troubleshooting
- Labnify's system will look for either "tracking" or "email" automatically. If another CNAME is used please contact support to get it validated.
- If you are using Cloudflare as your host, you need to disable it for that particular record by clicking the "cloud" icon.
- Some domain hosting providers add their own domain name at the end of a tracking record value automatically. So if you try to set a tracking record with value **relay.labnify.com**, it would be automatically set up as **relay.labnify.com.yourhostingproider.com** In order to work around this, just add a period "." at the end of your tracking record value, so it would look end up as **"relay.labnify.com."**.

4. DMARC
We strongly recommend using our DMARC Generator - it will help you create a DMARC record suited to your domain. However, since DMARC is not required, please note it will not be shown on the domains screen with other records. Having said that, you can still input the record in your DNS zone for additional layer of protection.

Domain-based Message Authentication, Reporting & Conformance is an email authentication protocol that is built on top of SPF and DKIM protocols. SPF and DKIM are prerequisites of DMARC and must be in place before setting up a DMARC policy.

A DMARC policy allows a sender to indicate that their emails are signed by SPF and DKIM and tells a receiver what to do if neither of those authentication methods passes – such as junk or bounce the email. DMARC removes the guesswork from the receiver's handling of these failed emails, limiting or eliminating the user's exposure to potentially fraudulent & harmful emails. DMARC also provides a way for the email receiver to report back to the sender about emails that pass and/or fail DMARC evaluation.

Note, there is no specific configuration needed in Labnify besides ensuring that your SPF and DKIM are both valid. Further, a DMARC policy on your domain(s) will affect all of your email sending from that domain (not just the mail you are sending through Labnify), so you need to ensure you are using SPF and DKIM for all your email delivery. For more information please click here.

The following are example DMARC TXT entries to set up on your domain(s) DNS.

Option A
Setup your DMARC policy with a simple, most common DMARC record. You will not receive any reports with this setup.
Host/Name:_dmarc
Value: v=DMARC1;p=none;

Option B
This setup will include reports. The DMARC Reports will come to the email you specify in ruf= and rua= parameters. If you do not wish to receive them anymore, remove these parameters (Similar to Option 1).
When you no longer receive negative reports, change your DMARC policy to quarantine which will not necessarily bounce email, but indicate to the recipient server they should consider quarantining it (junk or spam folder).
Host/Name:_dmarc
Value: v=DMARC1; p=quarantine; ruf=mailto:youremail@yourdomain.com;
rua=mailto:youremail@yourdomain.com
ruf - Forensic (failure) reports
rua - Aggregate reports

Option C
Another option with reports included. When you are satisfied that you are validating all the email from your domain(s) with SPF and DKIM, change the policy to reject which will bounce the emails that do not pass SPF and DKIM validation.
Host/Name:_dmarc
Value: v=DMARC1; p=reject; ruf=mailto:youremail@yourdomain.com;
rua=mailto:youremail@yourdomain.com
Please click here to view a list of the most popular tags available for your DMARC policy as above are only examples.

5. MX
For most users, you will not be adding or changing any MX records in your domain's DNS. The only reason why you would change or add an MX record for use with Labnify is if you are using Inbound Email Notifications which are webhooks that are part of our HTTP API. Otherwise, this will give you a green check mark if you have an existing MX record that is used for directing mail to your own mail server. If it does not give you a green check mark then you do not have any MX records for your domain. This is okay - it is not required and all it means is that you do not currently have a mail server setup for your domain.

Please note that new data input in a DNS zone of a domain can take up to 48h to propagate. It varies between hosting providers.

Frequently Asked Questions:
This section will help you with understanding more detailed obstacles that you might encounter during domain verification.

What are SPF Records? What are they for?

SPF stands for "Sender Policy Framework". An SPF record is in place to identify which mail servers are authorized to send mail for a given domain. It is used to prevent spammers from sending mail with fraudulent From addresses in that domain.

It is highly recommended that the SPF record is entered as a TXT record, as otherwise it will be not recognized as a valid record in most cases.
If you are creating the SPF record you will likely see a form with at least two fields, they are: "Host" or "Name" and "Value".
Here is what you enter under those fields:

Host/Name: @ (yes you put the @ symbol here under the host or name category)
Value: v=spf1 a mx include:relay.labnify.com ~all

SPF TIPS:

Check to see if there are any other SPF records in your domain's DNS. There can only be one SPF record per domain, so if there is an existing record just add "include:relay.labnify.com" to that record.
Make sure you remove the quotes.
For example, if your domain already has the record: v=spf1 a mx include:_spf.google.com ~all then you would just add: include:relay.labnify.com. The final record would look like this: v=spf1 a mx include:_spf.google.com include:relay.labnify.com ~all
Another tip: Sometimes '@' needs to be replaced with something else specific to your hosting. Check with your hosting provider to see if a special entry is needed in the Host/Name field if there is one. There are MANY different DNS editors and the set up can be different for each.

What is a TXT record?

This is a short form which means "text record" and can be nearly any form of text.

What are DKIM records? What are they for?

DKIM stands for "DomainKeys Identified Mail". They allow receiving servers to confirm that mail coming from a domain is authorized by the domain's administrators. This record also needs to be entered as a TXT record, you will see at least two fields, they are: "Host" or "Name" and "Value".

Here is what you enter in those fields:
Host/Name: api._domainkey
Value:
k=rsa;t=s;p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCbmGbQMzYeMvxwtNQoXN
0waGYaciuKx8mtMh5czguT4EZlJXuCt6V+l56mmt3t68FEX5JJ0q4ijG71BGoFRkl87uJi7LrQt1ZZ
mZCvrEII0YO4mp8sDLXC8g1aUAoi8TJgxq2MJqCaMyj5kAm3Fdy2tzftPCV/lbdiJqmBnWKjtwID
AQAB
(Yes, it really is this long list of numbers and letters, make sure you copy and paste correctly).
In some DNS settings, the Host/Name field may require you to enter
"api._domainkey.yourdomain.com", replacing your domain with your actual domain.

What is a "Verified Sender Domain"?

A verified sender domain is a domain that has given Labnify permission to send email from your domain. This means that the emails are being sent by Labnify servers, but the emails are using your domain as the "From" address.

Why is it a good idea to verify your sending domain?

Recipient ISPs need to verify that the domain has given permission for Labnify servers to use it. They can verify this by looking up certain records in your domain's DNS.

What does DNS mean? Can anything be a domain?

DNS stands for Domain Name System. Each Hosted Domain can be looked up by a server by it's domain or corresponding IP address. Each domain has its own settings. Nearly anything can be a domain, for example, myowncompany.com could be a domain if registered.

Where do I start, where do I go to fill in the correct form for this? What will I see?

First, you will go to where your domain is hosted (ie. GoDaddy, Dreamhost, Hostgator etc.), there will be an option there to adjust your DNS settings. You will see several fields of information to fill. The most common are SPF and DKIM.

Once I've entered those records in my DNS settings, what do I do next?

Once those records have been added to your domain's DNS settings, visit your Labnify account's Settings screen. Green Check marks mean that the domain records are valid and you can now send email from "anything@yourdomain.com".

What if the domain is set as the "Default"?

It is good practice to have a default verified sender domain. Our system will use the default verified sender domain if a non-verified domain is used for the email for the account. This is particularly important if sub-accounts are using the main account sender domain or resellers want notification emails to send using their domain.

What is a Tracking Domain?

If tracking is enabled for your account, then our system re-writes links in the emails. Any link will first direct the user through "relay.labnify.com".
So, for an instant, while the link is rerouted to its original destination, the URL will show "relay.labnify.com ".
We offer the ability to "whitelabel" this URL with your own domain. In your domain's DNS settings, create a new CNAME record:
Host/Name: tracking
Value: relay.labnify.com
Then go to your Labnify account's Settings Screen and click "Verify". Our system automatically looks for the CNAME "tracking". So if you enter anything else, please contact support so that we can validate the record manually. A green check mark means that the tracking domain is active.

How to verify your sender email

It is possible to verify your email address rather than the domain itself.

Why verify your sender domain?

Verifying your sender domain has multiple benefits - it proves that you are the actual owner of the domain and improves your delivery rate and reputation. It also removes the 500/day sending limit from your Labnify account.

Alternatively, if full domain verification is too technical or not possible for some reason, you can verify a single email address (or multiple) that you own. This process provides similar benefits to verifying your sender domain, but only for these particular addresses.

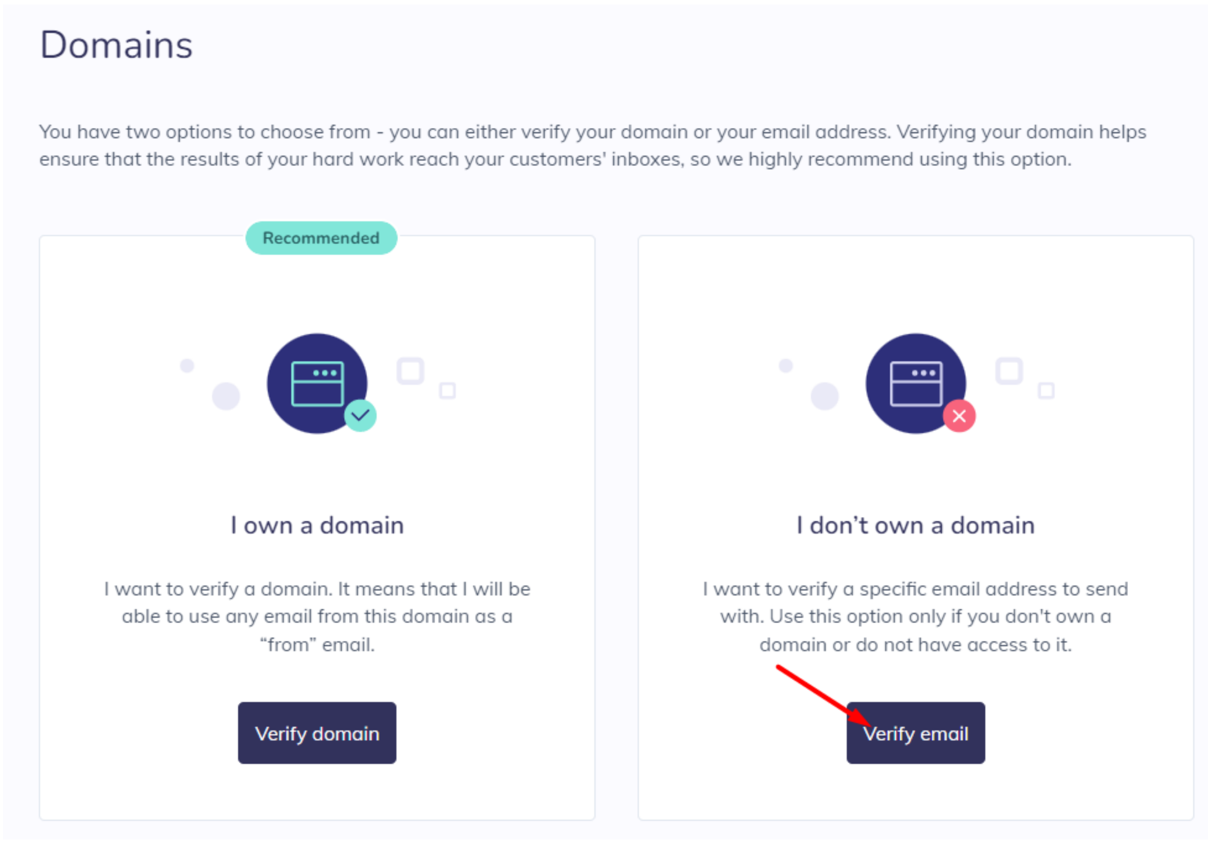
How to verify?

The email verification process is easy as the email address associated with your account should already be verified (this is the address that you’ve created the account with and where the activation link was sent).

If you'd like to add another email address:

Go to your Domain Settings screen on your Labnify account.

Choose **"Start Verification"** and pick the "**Verify email"** option.



Domains

You have two options to choose from - you can either verify your domain or your email address. Verifying your domain helps ensure that the results of your hard work reach your customers' inboxes, so we highly recommend using this option.

Recommended

I own a domain

I want to verify a domain. It means that I will be able to use any email from this domain as a "from" email.

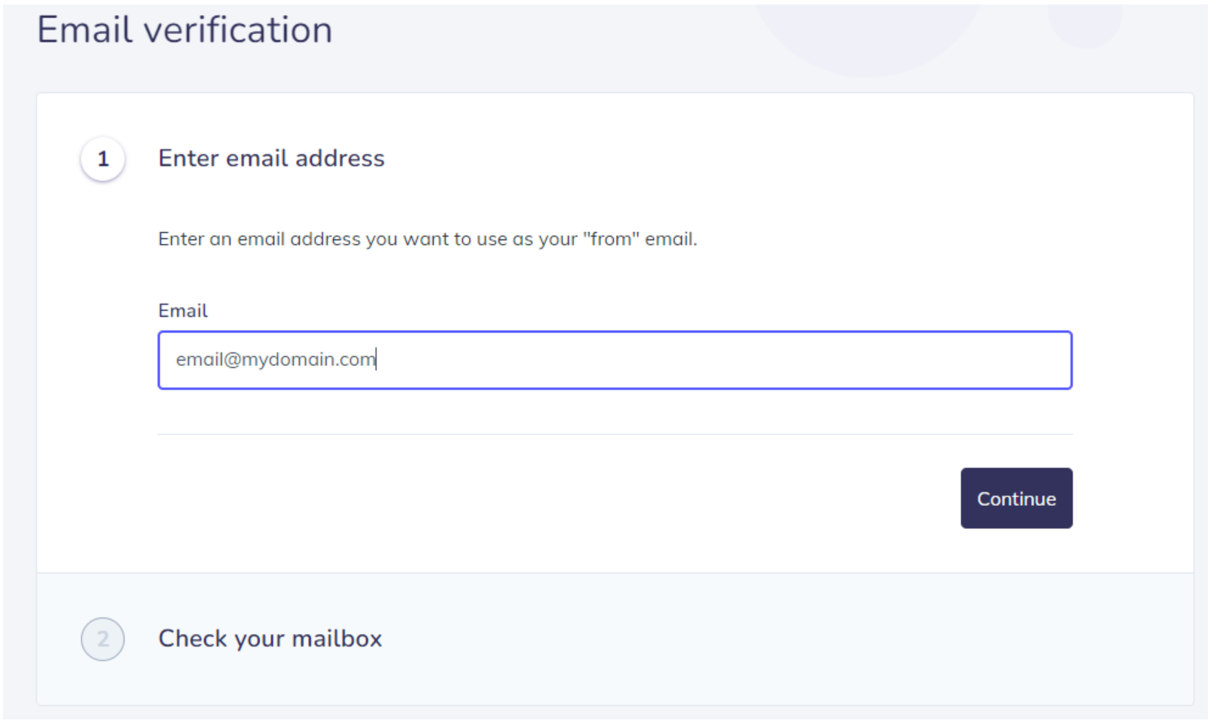
Verify domain

I don't own a domain

I want to verify a specific email address to send with. Use this option only if you don't own a domain or do not have access to it.

Verify email

Enter the email address and click **Continue**:


Email verification

1 Enter email address

Enter an email address you want to use as your "from" email.

Email

email@mydomain.com

Check your mailbox and click the confirmation link. If the email is not received, retry or check your other mailbox folders. If the email is still not received, please contact our Support Team.

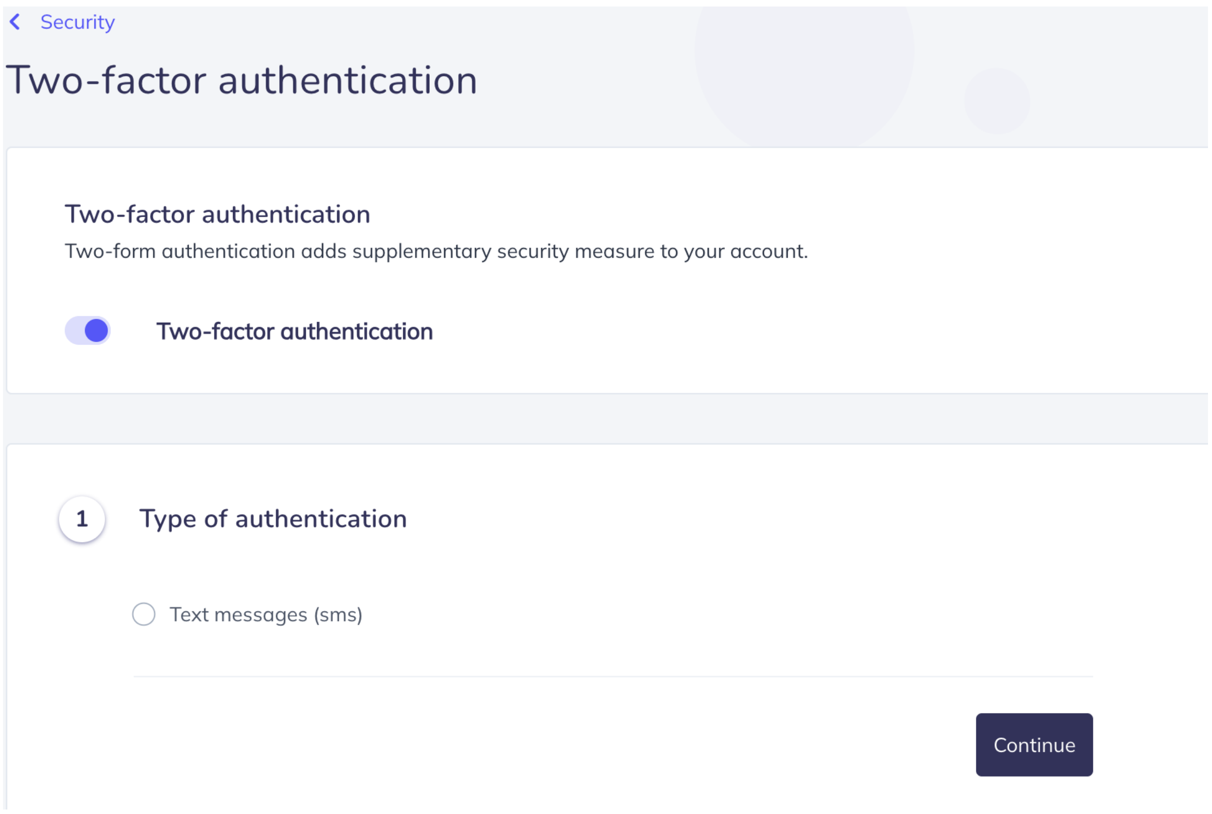
Done! Your email address is now verified!

You can do this with multiple email addresses. While this is a very simple and fast process, it is still recommended to fully verify your sender domain as it gives you more flexibility when choosing a sender address.

Securing your account with 2FA

Two-form authentication adds supplementary security measure to your account.

The **two-form authentication** (called **2FA** for short) is an additional feature that you can enable for your Labnify account's protection at any time on the security screen. There, you will find several options including the one below:


Security

Two-factor authentication

Two-factor authentication

Two-form authentication adds supplementary security measure to your account.

Two-factor authentication

1 Type of authentication

Text messages (sms)

When you click "**enable authentication**", you will be able to verify using Text message.

A **security code** will be sent to your **mobile** each time you (or someone else) attempt to log into your Labnify account using the classic credentials (email, password). Entering the security code will **grant access** to the account.

(**Note!** Make sure that your cellular service is within reach and/or your device stays connected to Wi-Fi (or has enabled mobile data) in order to properly receive the security code.)

How to set your domain as a default

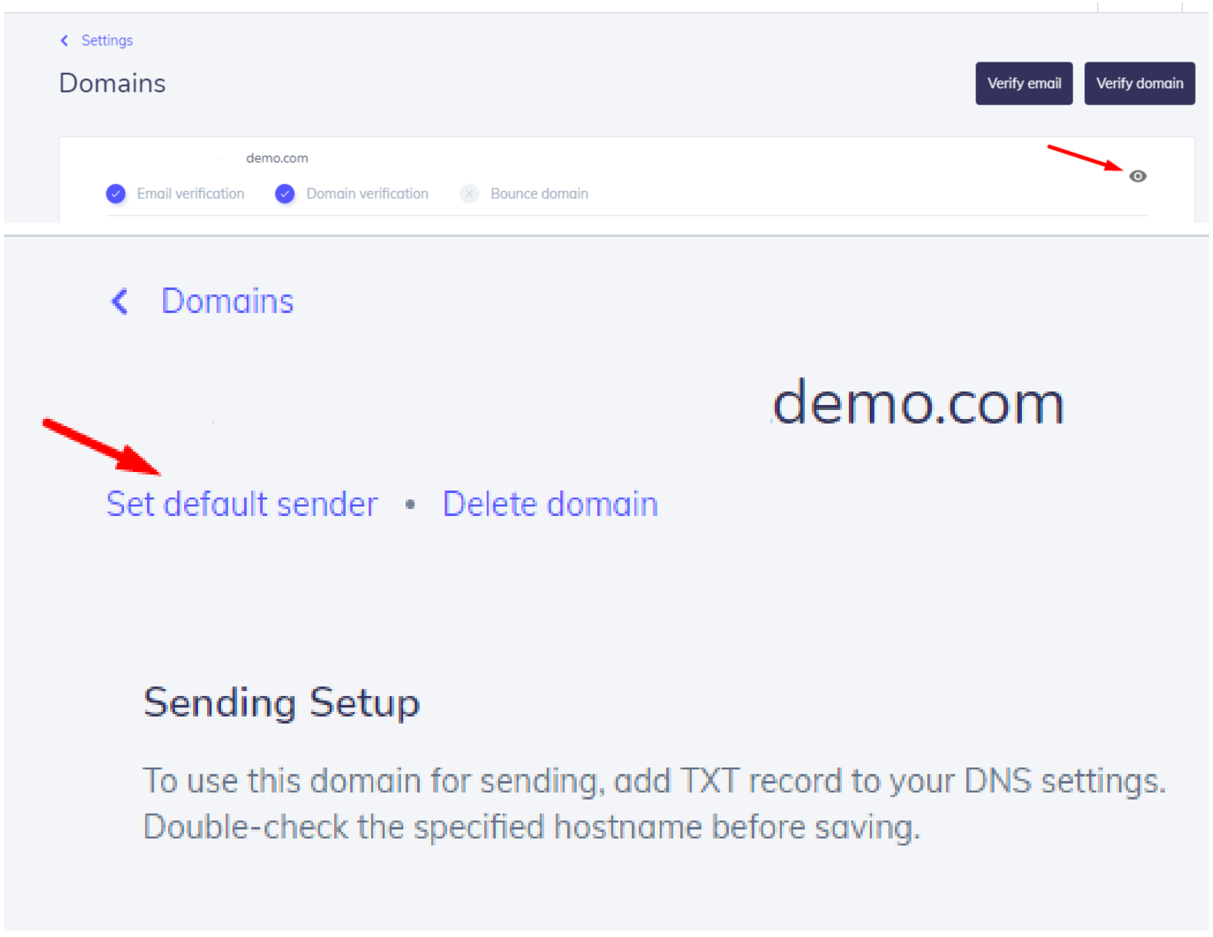
Setting a domain as the default sender is important for your account

What does it mean to have the domain set as the "Default" sender? It is good practice to have a default verified sender domain. When there is a default domain set on the account, emails are still branded using the default domain, even if sent from a non-verified domain.

If there is no default domain set, and an email is sent from a non-verified domain, then labnify.com or labnify.net is used. (please note: you have to verify your domain before you can be allowed to send emails).

An account with a default sender domain also has increased deliverability performance as some inboxes do not recognize accounts without this set legitimate senders.

You can set your domain as the default sender on your Settings>Domains screen:



Sending Setup

To use this domain for sending, add TXT record to your DNS settings. Double-check the specified hostname before saving.

You will know the domain has been set as the default successfully when a star appears on the left of the domain name.

Setting up SPF & DKIM

Understanding why SPF and DKIM records are important for your success is crucial to ensure best possible deliverability of your emails. Both these records are set up in your **DNS Zone** - the administrative space on your hosting where domains and their records are managed.

DNS Zones differ a bit depending on what hosting they belong to, so if you encounter any problems with setting up these records, please contact your hosting provider.

Sender Policy Framework

SPF (Sender Policy Framework) is an authentication protocol. It helps recipient servers to identify you as a legitimate sender and owner of your FROM address. Setting up a correct SPF record will prevent other people from sending emails on your behalf. Simply put, it allows you to specify who will be allowed to send emails on behalf of your domain. If you aim for best email deliverability, SPF is absolutely essential.

It is also set up in order to stop phishing attacks. SPF record tells email servers if an email was sent from authorized sender IP address. Thanks to this information, server administrators can easily detect and block phishing emails.

Labnify SPF record looks as follows. Below you will find a breakdown of its syntax:

```
v=spf1 a mx include:relay.labnify.com ~all
```

"v=spf1" - this simply indicates the version of SPF record. Current standard is always version 1

a - a test for the A record of the domain. In order to pass, it should match the sender IP.

mx - a test for the MX record of the domain. In order to pass, it should match the sender IP.

include - The third-party domain that is defined after the "include" directive (in this case it would be relay.labnify.com) is allowed to send on your behalf.

~all - This directive indicates that any other IPs or domains than the ones specified in the record are not authorized to send on your behalf and will soft-fail.

DomainKeys Identified Mail

DKIM (DomainKeys Identified Mail) allows receiving servers to confirm that mail coming from a domain is authorized by the domain's administrators.

It is achieved thanks to a pair of cryptographic keys – one is a public key published in a TXT record and the other is a private key encrypted in a signature affixed to outgoing messages. Both are generated by Labnify.

DKIM is not as essential as SPF but having your emails signed with DKIM will further help recipient servers with treating you as a legitimate sender. It further improves the chances of your emails not landing in your recipient's spam folder.

Correctly set up DKIM record will also make it harder to spoof your emails for any malicious third party.

An example of a decrypted Labnify DKIM record (as you will see it in the email headers after receiving it in your inbox) looks as follows.

Below you will find a breakdown of its syntax:

```
DKIM-Signature: v=1; a=rsa-sha256; d=mydomain.com; s=api; c=relaxed/simple;
t=1657630312; h=from:date:subject:reply-to:to:mime-version;
bh=WP7kmx0OyB67VOSbecKqjSAS/xemAzmsmWxeqZCvzfU=;
b=jGwN8zV3F/KfjxArDVQIe9NT7k2Hrf68w041Wwd11L6WyXn2lypT4UXH+sQjr7l+2/heM2lYt
24
```

FaqzLOZKtjAXZCyCU5GeJHkZiUxkd1NO6ARYESncklONZKdfxxlx1LN7QY16HZDPWJY6hYH7Vwf
fF V/CtBfGOanpaqak+lNA=

DKIM-Signature - a header indicating the beginning of DKIM entry

v=1 - the version of DKIM used by the sender

a=rsa-sha256 - the algorithm used to generate the hash for the public and private keys.

c=relaxed/simple - canonicalization posture for the sending domain. In simple terms, it regulates whitespace and text wrapping changes that may occur in the email.

s=api - selector for the public DKIM key used when verifying it. A domain can have multiple DKIM keys. The role of the selector is to make sure that recipient servers use the right public key.

d=mydomain.com - domain used when signing the email message. With the current version of Labnify DKIM, relay.labnify.com should be used here.

h=from:date:subject:reply-to:to:mime-version; - headers included in the message when it was signed

bh=WP7kmx0OyB67VOSbecKqjSAS/xemAzmsmWxeqZCvzfU= - The value of a body hash that's generated before the headers are signed.

**b=jGwN8zV3F/KfjxArDVQIe9NT7k2Hrf68w041Wwd11L6WyXn2lypT4UXH+sQjr7l+2/heM2I
Yt24
FaqzLOZKtjAXZCyCU5GeJHkZiUxkd1NO6ARYESncklONZKdfxxlx1LN7QY16HZDPWJY6hYH7V
wffF V/CtBfGOanpaqak+lNA=** - The cryptographic signature of all the previous information from the DKIM-Signature field.

t= this tag indicates that the domain is testing DKIM or is requiring a domain match in the signature header between the "i=" and "d=" tags.

Adding DMARC record

Domain-based Message Authentication, Reporting & Conformance is an email authentication protocol that is built on top of SPF and DKIM protocols. SPF and DKIM are prerequisites of DMARC and must be in place before setting up a DMARC policy.

A DMARC policy allows a sender to indicate that their emails are signed by SPF and DKIM and tells a receiver what to do if neither of those authentication methods passes – such as junk or bounce the email. DMARC removes the guesswork from the receiver's handling of these failed emails, limiting or eliminating the user's exposure to potentially fraudulent & harmful emails.

In short, this basically means that a valid DMARC record is treated as a kind of certificate of being a **legitimate sender** to the recipient's server.

DMARC also provides a way for the email receiver to report back to the sender about emails that pass and/or fail DMARC evaluation.

Note there is no specific configuration needed in Labnify besides ensuring that your SPF and DKIM are both valid.

Further, a DMARC policy on your domain(s) will affect all of your email sending from that domain (not just the mail you are sending through Labnify) so you need to ensure you are using SPF and DKIM for all your email delivery. For more information please click here.

The following are example DMARC TXT entries to set up on your domain(s) DNS.

Step 1 - Setup your DMARC policy to simply notify you of mail that is not passing SPF and DKIM
Host/Name:_dmarc
Value:
v=DMARC1;p=none;pct=100;rua=mailto:youremailaddress@yourdomain.com;ruf=mailto:youremailaddress@yourdomain.com

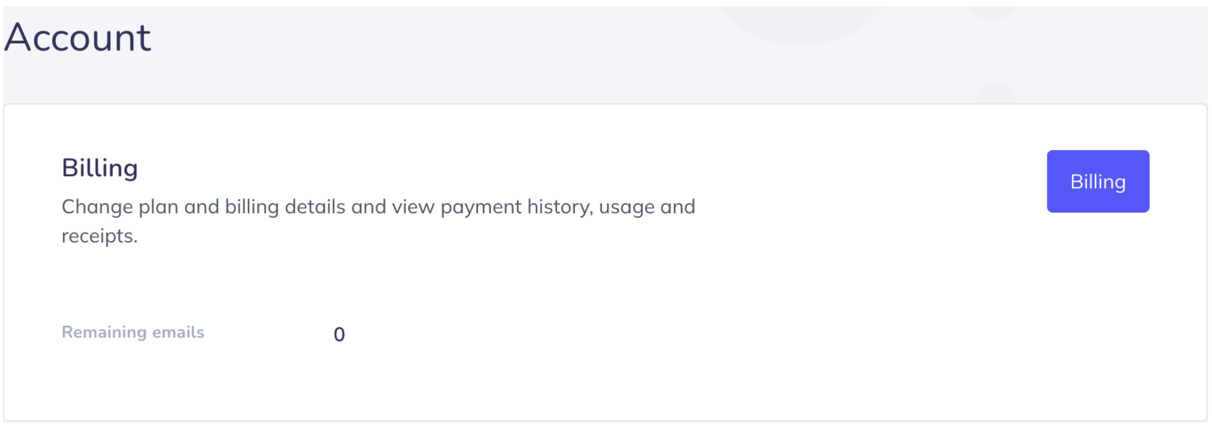
Step 2 - When you are no longer receiving negative reports, change your DMARC policy to quarantine which will not necessarily bounce email, but indicate to the recipient server they should consider quarantining it (junk or spam folder).
Host/Name:_dmarc
Value:
v=DMARC1;p=quarantine;pct=100;rua=mailto:youremailaddress@yourdomain.com;ruf=mailto:youremailaddress@yourdomain.com

Step 3 - When you are satisfied that you are validating all the email from your domain(s) with SPF and DKIM change the policy to reject which will bounce the emails that do not pass SPF and DKIM validation.
Host/Name:_dmarc
Value:
v=DMARC1;p=reject;pct=100;rua=mailto:youremailaddress@yourdomain.com;ruf=mailto:youremailaddress@yourdomain.com

Please click here to view a list of the most popular tags available for your DMARC policy as above are only examples.

How to manage your billing

The Billing screen is a vital part of your account. As soon as you decide to purchase a paid plan, familiarity with it will surely help.


Account

Billing

Change plan and billing details and view payment history, usage and receipts.

Remaining emails 0

You can access your billing preferences under the Billing section of your account menu.

Here you can set your preferred payment option and plan that suits your need. You can connect with our support via Live Chat under Help section if you have any queries.

Email Magic product

Everything you need to know to start using our Email Magic platform. Find out what you need to do to manage your contacts, templates, landing pages and send your first campaign.

How to upload your contacts

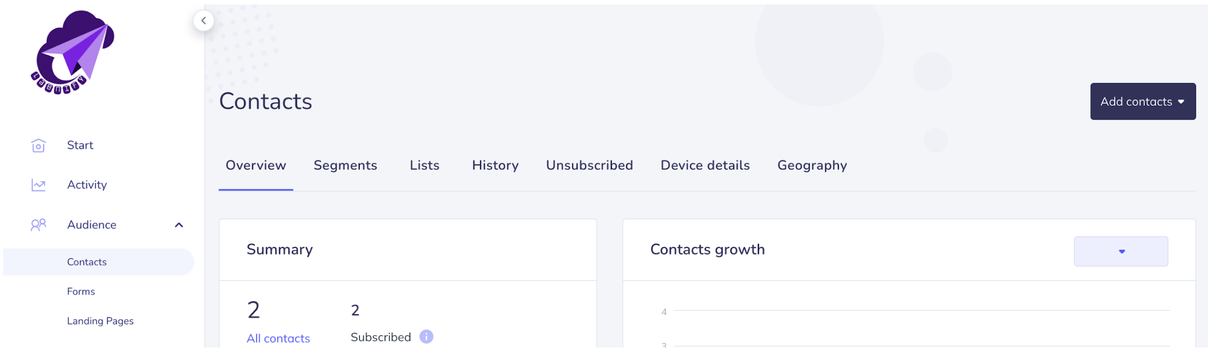
There are multiple ways of adding contacts to your account and managing them.

Adding contacts - via .CSV file upload
Please note that the .csv file you upload has to be in **UTF-8** encoding format. Making sure the encoding of your file is correct will guarantee the .csv file is correctly recognized and formatted.

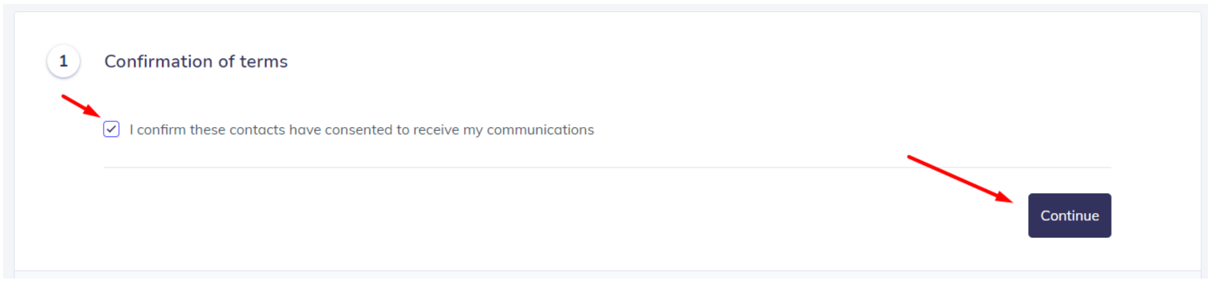
Proper CSV format:
CSV file stands for **"comma delimited values"** which are values separated by a comma, just as in the following example:

```
email,firstname,lastname
mycomapny@domain.com,John,Doe
```

Navigate to your **Audience** menu and select "Upload Contacts" from the "Add Contacts" drop down.

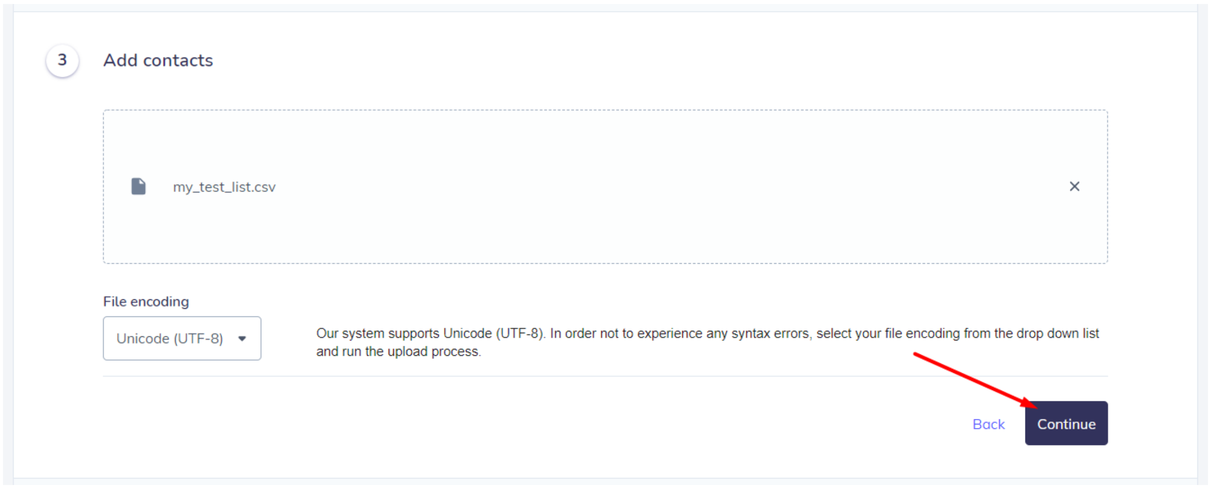


1. Confirm your contact's consent.


1
Confirmation of terms
I confirm these contacts have consented to receive my communications
Continue

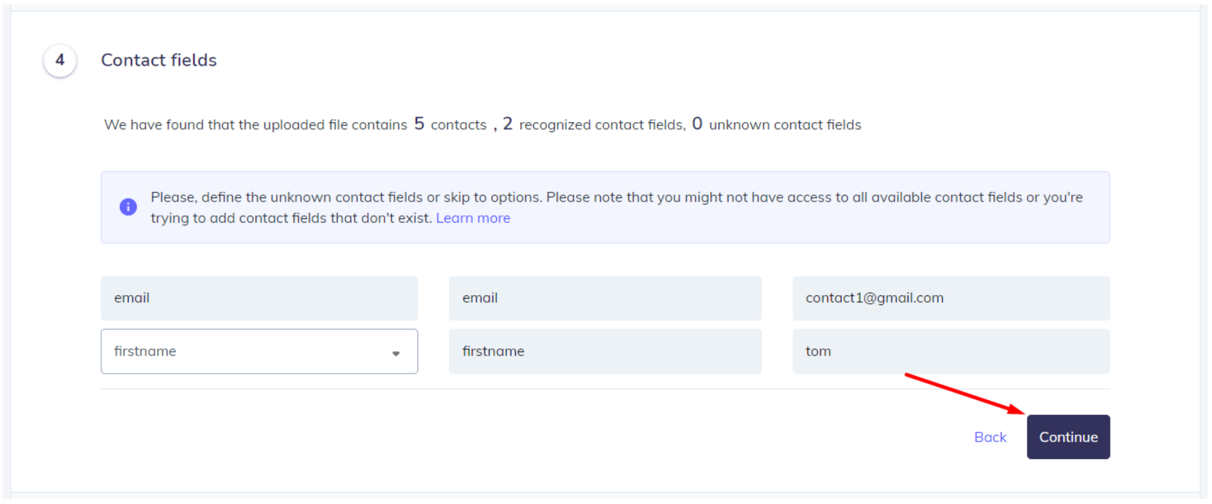
2. Choose the "Upload contacts from file (zip, csv, json, xml)" and simply drag and drop the .csv file onto the screen.

The upload tool supports a single .csv file or a .csv list compressed into a .zip file. It is recommended that you keep the file size to a maximum of 25MB.



Our system supports Unicode (UTF-8). In order not to experience any syntax errors, select your file encoding from the drop down list and run the upload process.

3. You can then customize the contact fields.


4 Contact fields

We have found that the uploaded file contains 5 contacts , 2 recognized contact fields, 0 unknown contact fields

Please, define the unknown contact fields or skip to options. Please note that you might not have access to all available contact fields or you're trying to add contact fields that don't exist. Learn more

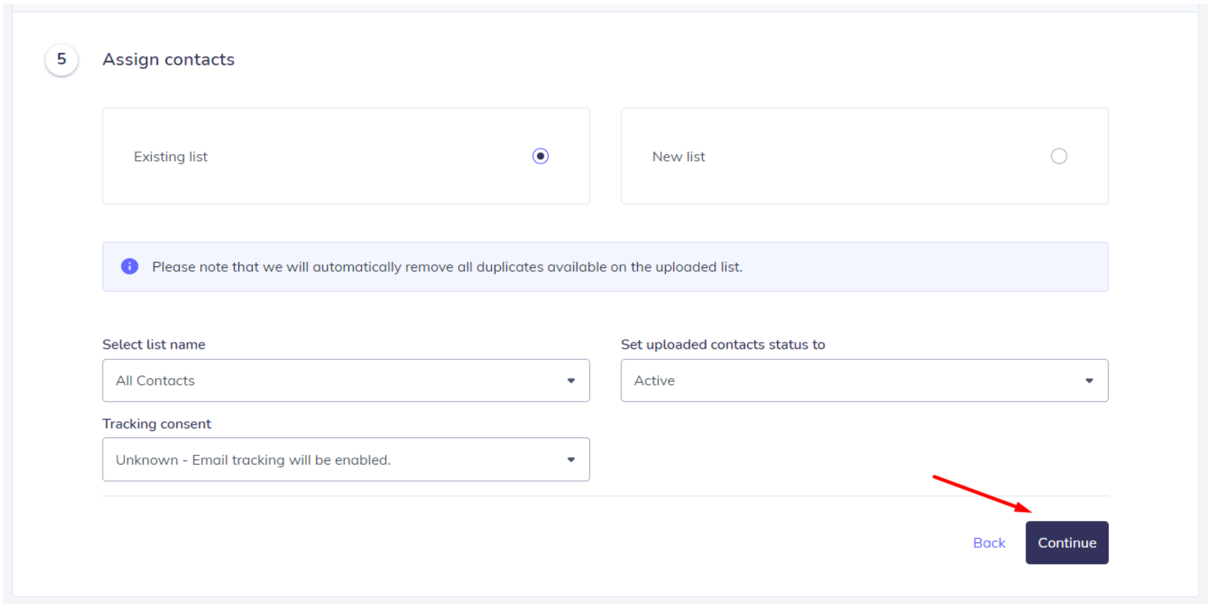
email	email	contact1@gmail.com
firstname	firstname	tom

Make sure there is exactly one field labeled 'email'. There cannot be multiple contact field labels that use 'email'. The system will auto-detect existing contact fields as well as the following:
firstname, lastname

Here are some example values for these fields:
someone@example.com, John, Doe,
someone2@example.com, Jane, Smith,

At the minimum the .csv file can simply be a column of email addresses:
email
address1@domain.com
address2@domain.com
4. Then choose to add the contacts to either an existing static list or create a new static list for the contacts to be added to. This is also the step during which you can choose the final status of your uploaded contacts.

Larger files may take some time to process.


5 Assign contacts

Existing list [✓] New list

Please note that we will automatically remove all duplicates available on the uploaded list.

Select list name: All Contacts
Set uploaded contacts status to: Active
Tracking consent: Unknown - Email tracking will be enabled.

Tips:

- Your column names **cannot contain spaces**. So a column called "first name" instead of "firstname" will not be accepted by our system.
- Remember not to add any characters or white spaces to the basic contact fields in the file - ***email, firstname, lastname***. With any extra characters they will not be recognized
- If you have a Microsoft Excel file you can save the file to .csv by choosing Save As and choosing the .csv extension.
- If you are using a Mac computer, you will need to convert the file to CSV before uploading it.
- List and segment names have a limit of 128 characters

Adding contacts - via HTTP API and SMTP Relay
Contacts can also be added to the account using our API.
Additionally, all recipients that are sent an email via API and via the SMTP Relay are automatically added to the account as a contact.

Contact List - Upload Filtering
When you upload contacts, on the last step, you are provided with information about how many contacts were uploaded as Active, Inactive, Invalid and Unsubscribed and why they were uploaded as such.

Managing your contacts

Knowing how to properly manage your contacts can make or break your email campaigns.

Your contacts are arguably the most important part of your email marketing strategy. Without your subscribers, you wouldn't have an audience for your email campaigns. Knowing how to manage them properly can take some practice and requires the right set of tools. Labnify has those tools and the expert advice you need to manage your contacts effectively.

Segments vs Static Lists

"Lists" are static sets of contacts while "Segments" are sets of contacts that dynamically change based on the segment's query.

An example is when a .csv file is uploaded, it is added to a new or existing list. That particular set of contacts does not change and remains static under that list.

A segment is completely dynamic and only ever contains the contacts that match the segments' Query. A simple example is the "Engaged Contacts" segment which is available by default on most accounts. This segment only ever contains contacts with an Engaged status (the contact has clicked or opened your email within the last 6 months).

This is what the query looks like Query: Status = 'Engaged'.

This is an extremely powerful tool that allows sophisticated segmenting of contacts based on a wide range of attributes.

Statistics
The new contact screen combines a set of different statistics related to your contact engagement which may help you plan your future campaigns and makes contact management easier.

Summary
Your contacts summed up and divided into all contacts, contacts with deliverable statuses and contacts that were added today.

Contact growth
A graph presenting the changes in the number of your contacts. You can adjust the date range of the graph.

Recent growth
The most recent (last 30 days) changes to your contact number, including unsubscribes.

Audience performance
Engagement of your contacts broken down into the percentage of opens and clicks.
Contact Lists - Creating New List

1. Go to your Contacts Screen. Click on "Create List".
2. New List: Give your list a name.
3. Allow Unsubscribe: When enabled this setting works with the {unsubscribelist} merge field to allow recipients to see which lists they are on.

TIP! If Allow Unsubscribe is enabled then your recipients can see the list name. Give the list a name that means something to both you and your contacts.

Contact History
The history tab will allow you to select and view segment counts on a graph as a function of time. Because segments are dynamic, they can change over time. The graph gives a visual representation of the segment changes over the last 30 days.

Web Form
Using a web form to gather subscribers is one of the ways to add contacts to the account.

Randomized Label Tool
The "Randomized List" tool will create a specific number of random contacts from any list or segment.
The tool will allow you to specify the name of the new list being created and the number of random contacts to be added to it from the list or segment which is currently selected.
There is an option to "exclude blocked contacts" if you only want Active and Engaged contacts on the new list.

Nth Selection Label Tool
The "Nth Selection" tool is very handy if you need to evenly break up an existing list or segment.
The tool will evenly distribute contacts to a specific number of different lists. The tool will allow you to specify the name of the 'List Series' and specify the number of lists to be created.

Statuses
"Show Contacts" will view the Label or List directly and display the first 20 pages of contacts. The "Status" drop-down offers status filters that show only the contacts with the particular status(es) that is selected.

Other Tools
For any list or segment, you have the option to export contacts, copy contacts to other lists, or delete contacts from the account.

For lists specifically, there is an option to simply remove contacts from that list but not delete them from the account.

When viewing the contacts for a particular list or segment you can use the "Select All" option which will select all the contacts in that list or segment that also match the status filters currently applied and will provide a modal of options that are available for those contacts.

Designing a template

A good template can be used again and again. Become familiar with your templates screen and all that it has to offer.

Labnify supports both body_text and body_html in our emails. Templates in HTML can be used over and over in your emails.

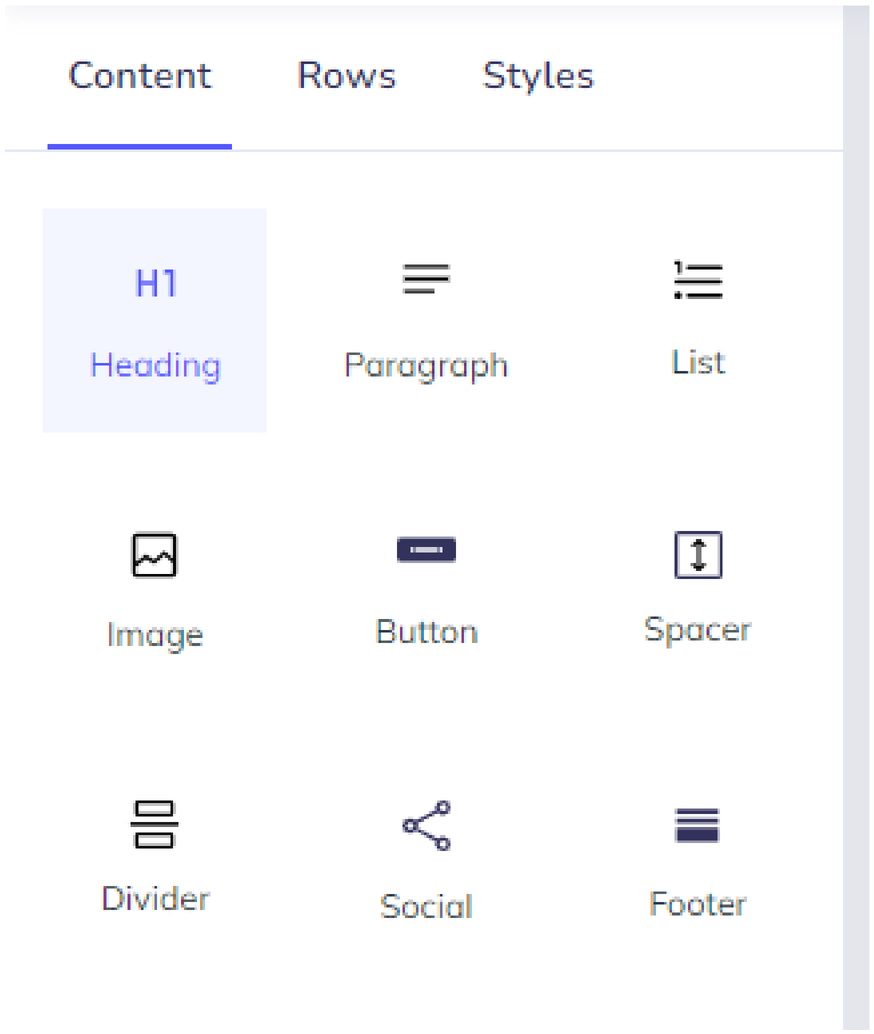
My Templates
The "Templates" tab will display templates saved in your account. By clicking on the "Preview" button under the three-dot burger menu of a template, you can preview any template and see how it might generally appear on any desktop or mobile device.

Gallery
We offer a gallery of pre-designed templates that you can customize. To use a free pre-designed template from the Gallery, just click the desired template which will auto-save the template and place it in the "My Templates" section.

Editors
If you want to create a template from scratch, you can do so by choosing "Create Template", and then choose either "Email designer", or "Raw HTML".

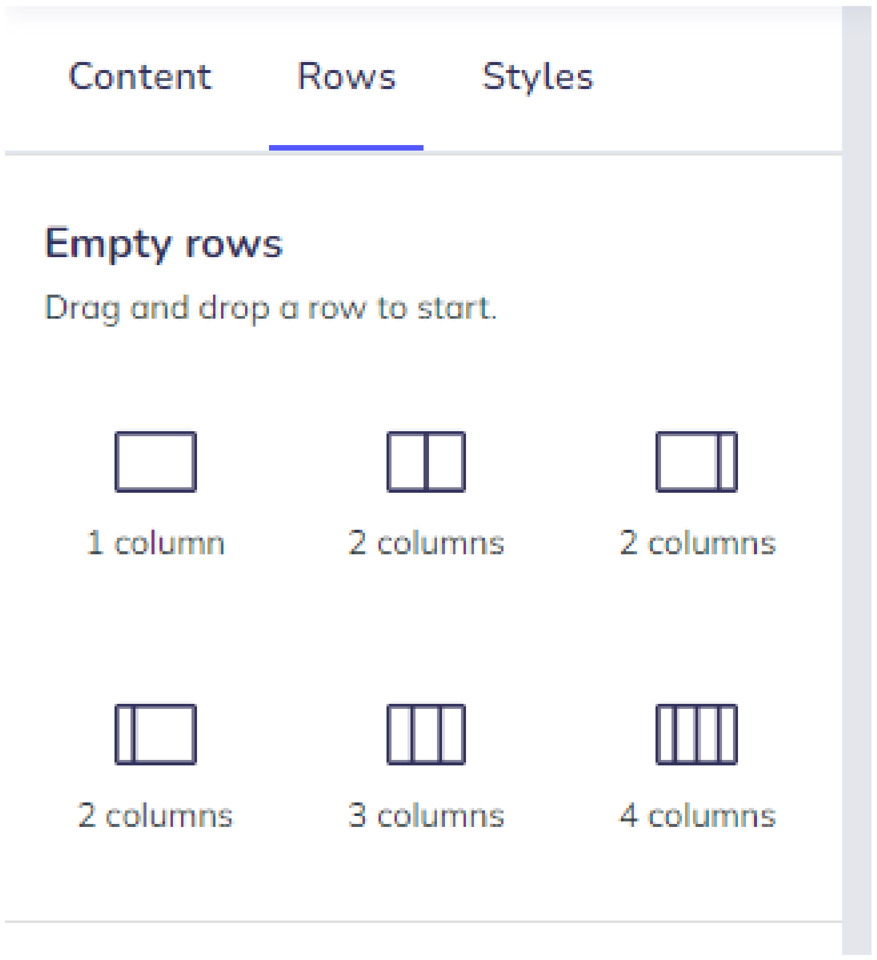
Email designer
In this Email Designer you are free to build your email template entirely from scratch using the available building blocks. The **default options** will allow you to set a Template Name and attach a default From Email Address, From Name, and Subject.

Content
The "Content" tab lets you customize general text, links header styles and other "blocks" available in creating the template's general design:



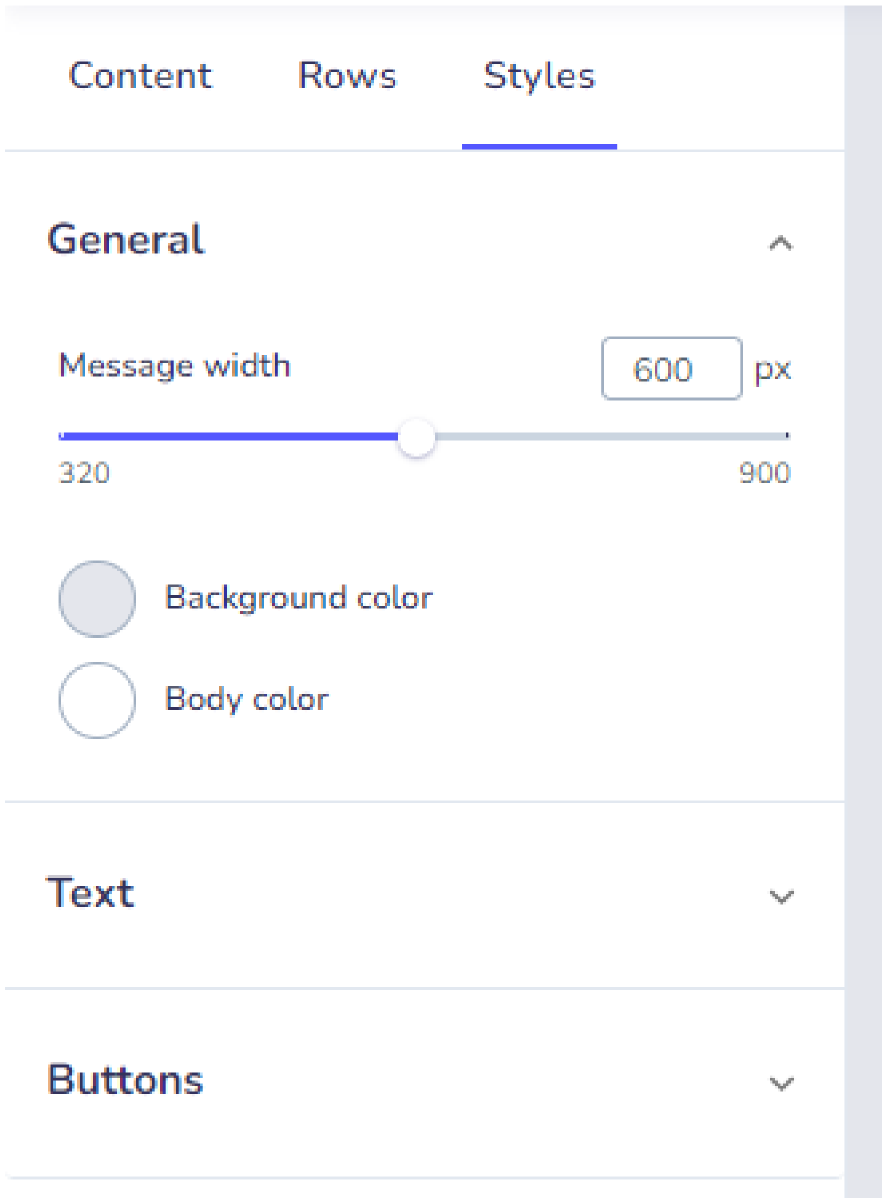
- Headings: Bigger piece of text to begin your paragraph with
- Paragraphs: Separate paragraph of text
- Lists: A simple yet elegant list - bulleted or numbered
- Images: Images previously uploaded to your media manager (supported image extensions: PNG, JPG, GIF)
- Buttons: Customizable and clickable buttons
- Spacer: Horizontal piece of empty space dividing parts of your email
- Divider: Horizontal line dividing parts of your email
- Social: Icons linking to your Social Media pages
- Footer: customizable text that will appear that the bottom of your emails

Rows
The "Rows" tab contains layout boxes that can be dragged over and dropped into the template. After placing your rows into a desired shape, you can then insert additional content boxes into the parts of the rows themselves.



- 1 column
- 2 columns with one bigger on the left or right
- 3 columns
- 4 columns

Styles
The "Styles" tab allows you to change the styles like fonts colors and sizes of existing template parts like buttons and links. Our Email Designer creates fully customizable and responsive templates.


Content Rows Styles

General

Message width 600 px

320 900

Background color

Body color

Text

Buttons

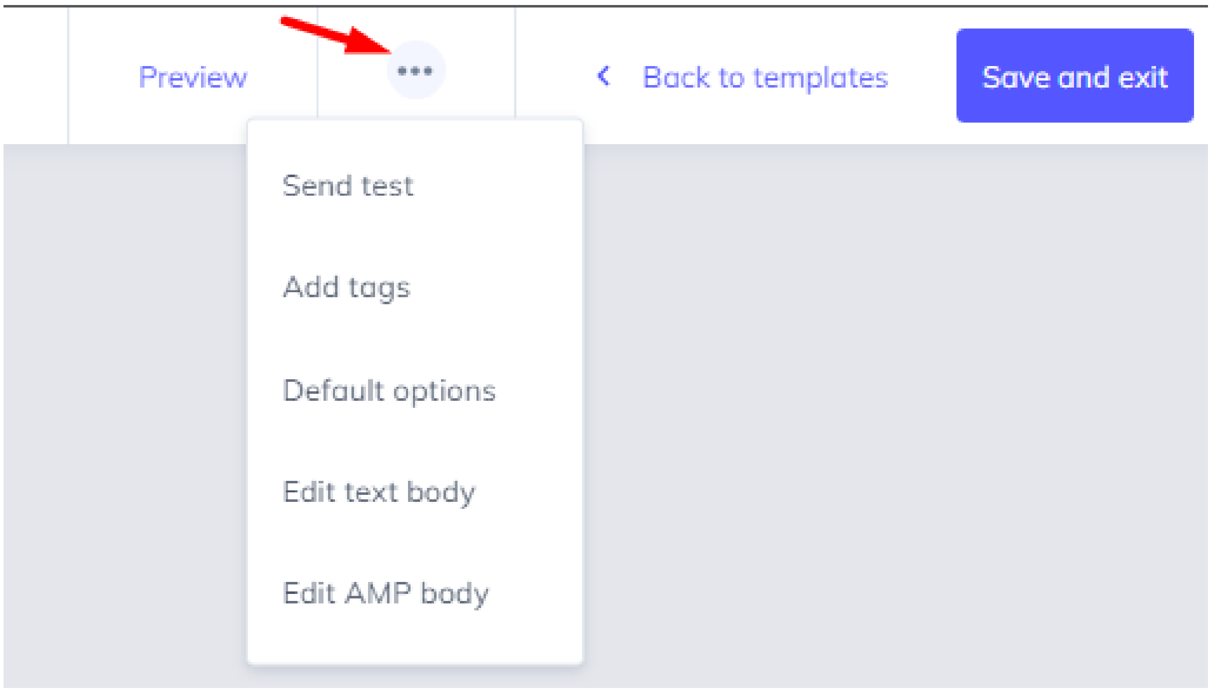
Customizable parts include:

- Message width
- Background color
- Body color
- Text size, colors and fonts
- Links
- Button color, font and radius

Preview

The "Preview" button will show you what your template will look like on an actual device, with the option of choosing a mobile preview too.

"More options" menu
The "more options" menu will allow you to set up final touches to your template



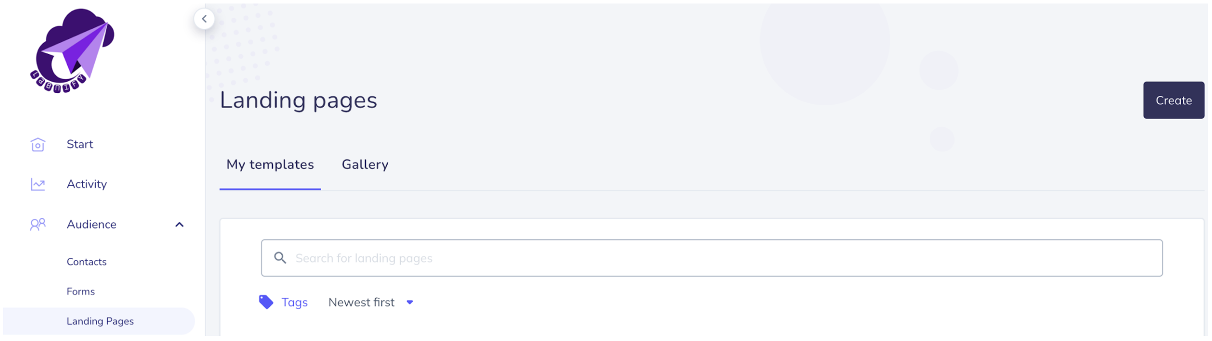
- Send test: Allows you to send a single test email to see how the email will look in an actual inbox
- Add tags: Add tags to this template in order to organize and search for it in your gallery
- Default options: Set up a default from email, from name and subject
- Edit text body: allows you to provide a body_text for the email in case a recipient mail server only accepts body_text. Your account has a setting that is on by default to auto-generate a text body from an HTML template. You can find this setting in your Advanced settings
- Edit AMP body: allows you to provide AMP body into your email. It will be accessible for every recipient whose inbox is compatible with AMP emails. It is entirely optional and can be ignored if you do not plan to send such emails.

Managing a landing page

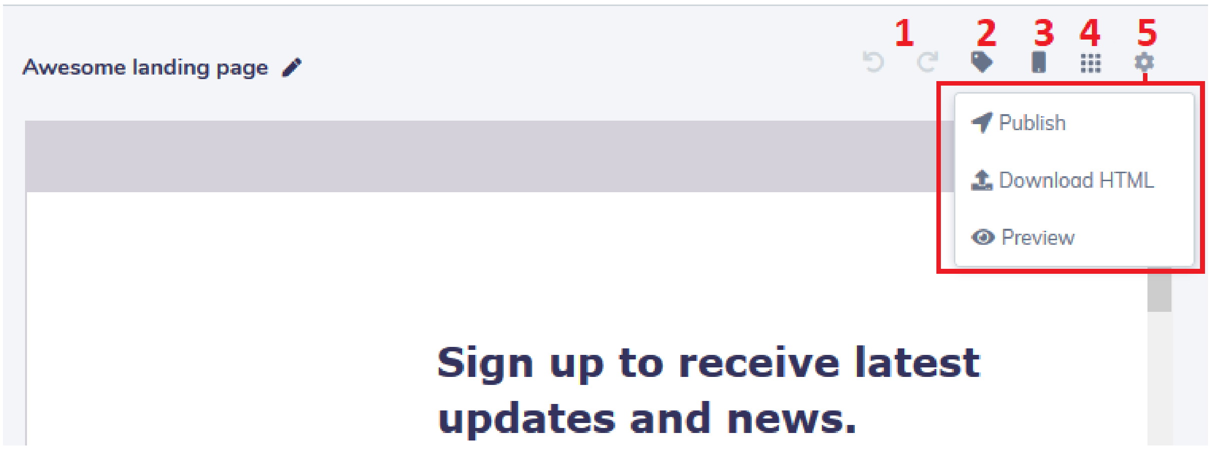
Learn how to create and manage a landing page in our easy to use drag and drop editor.

A landing page is a simple destination page that your recipient lands on when they decide to interact with your newsletters by clicking links or ads. You can access the Landing Page Creator just below the Templates section of your dashboard.

Creating a landing page
In order to start working on a new landing page, just navigate to **Audience/Landing Pages** screen.



All changes made in the editor are saved automatically.



1. Navigation buttons - They allow to undo and redo any changes made to the landing page. Move between previous and next action taken with ease.
2. Tag - Here you can assign a tag to your landing page. Tags can be later managed on the Templates screen when you are finished with the editor.
3. Mobile view - Our editor is highly reactive, here it can show you in real-time how your landing page will look on a mobile device

4. Grid - This button turns on the visual grid that shows boundaries of all elements of the landing page
5: Options>Publish - Clicking this button will show a pop-up window with a link to your landing page ready to be published anywhere you wish.
Options>Download HTML - Gives you the ability to download an HTML version of your landing page for personal use and export purposes.
Options>Preview - Preview your landing page and how it will look when published in the current state. Here you can also switch between mobile and desktop views.

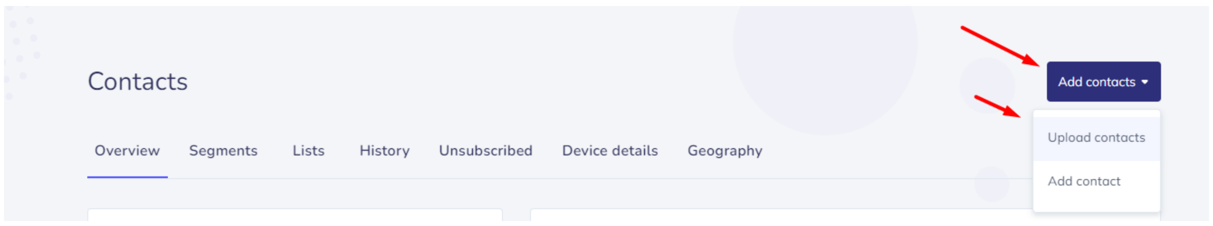
On the right there are blocks you can use to design your perfect landing page. With their help you can create and separate text, images, buttons and social media icons. Clicking the bottom-right corner of each block will present you with their different configuration options. Additionally, you can add one of your **double opt-in web forms**, if there are some previously created on your account. If not, just go to the **Forms** page and design one.

Sending your first campaign

This guide will walk you through the process of sending your first campaign.

When using Labnify, when you send an email to any group of contacts we call that a "campaign". To send a campaign you need a template (which becomes the email body itself) and you need contacts (the recipients who receive the email). Let's start with contacts.

The easiest way to add large groups of contacts to your account at once is to upload a .csv file:

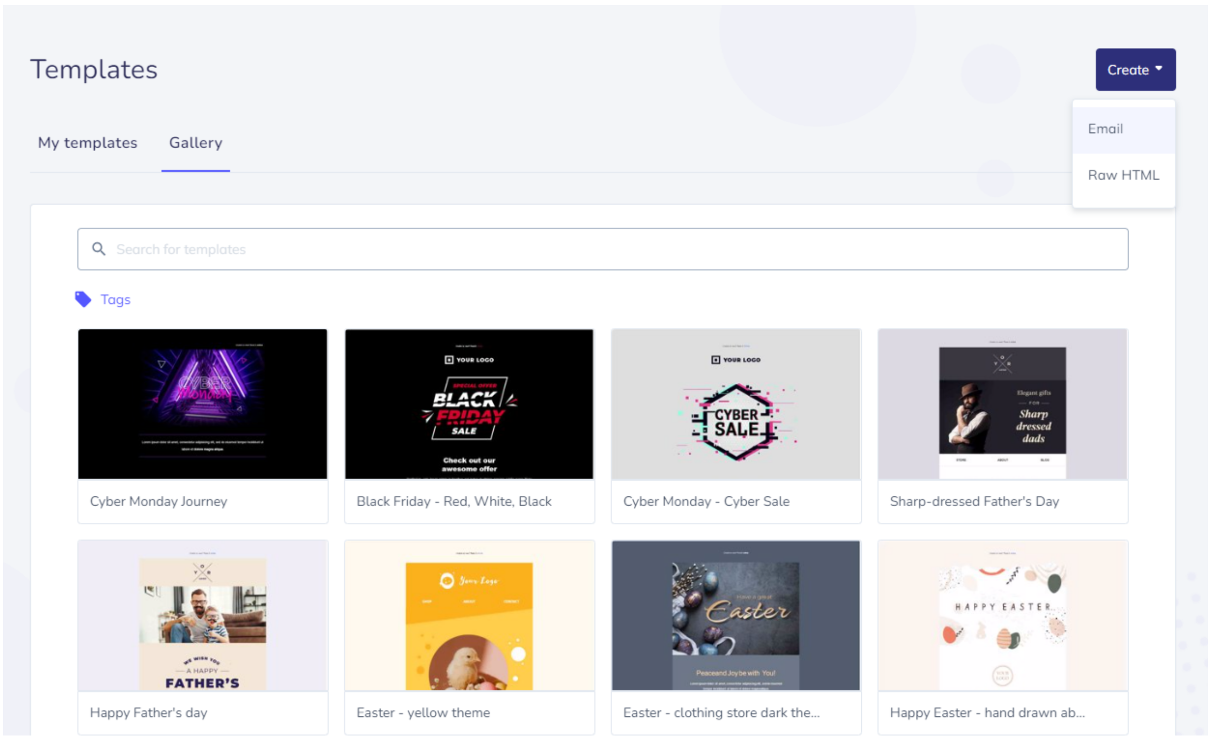


Most spreadsheet software should allow you to export your contact list as a .csv file.
Make sure your .csv file has a minimum of at least one column labeled 'email' that contains a list of email addresses.

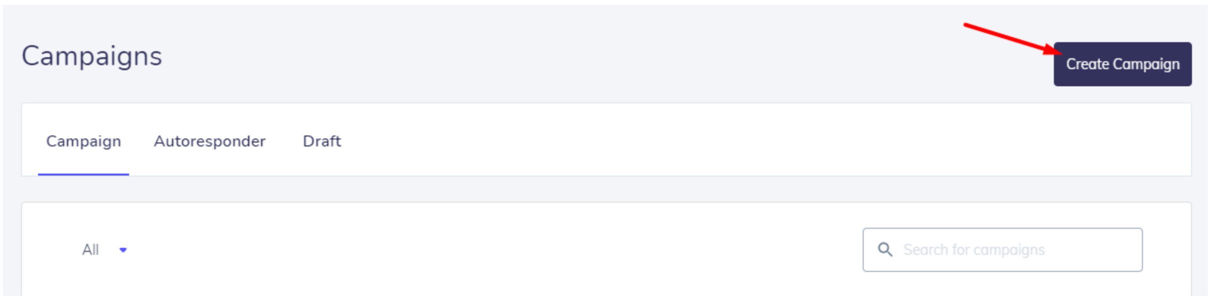
Example:

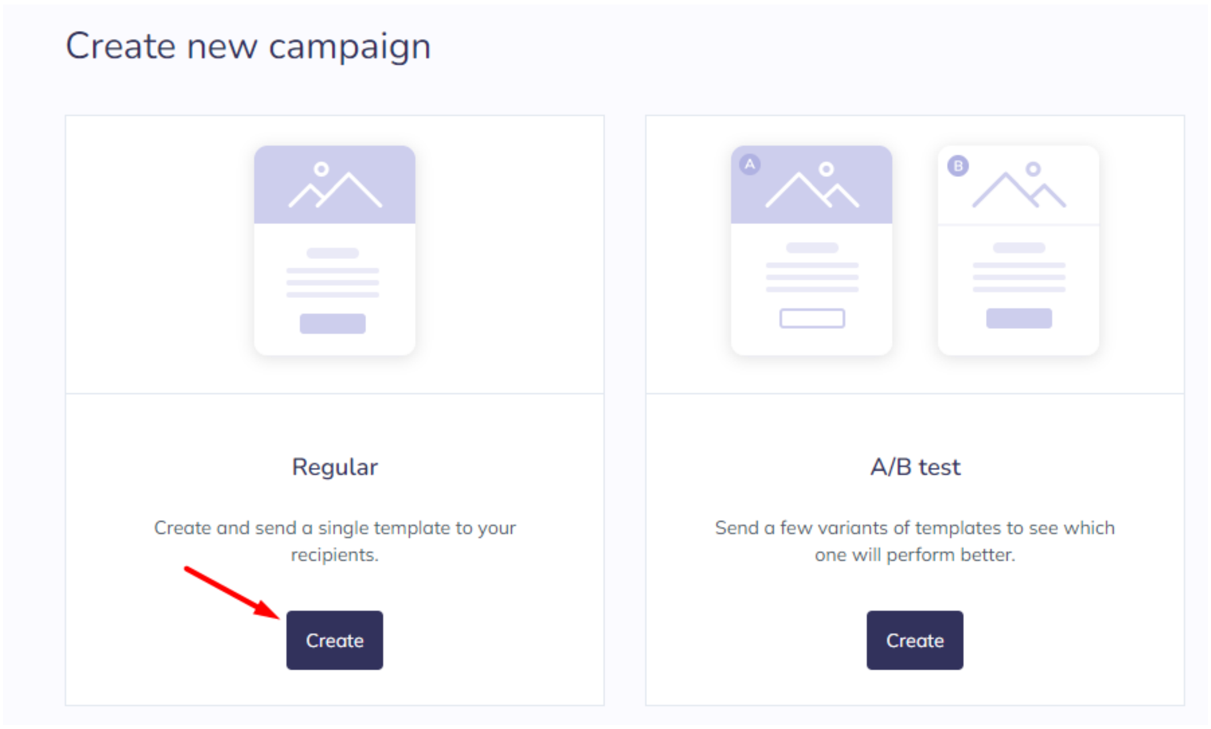
email
company@yourdomain.com

Now that you have contacts in your account, it is time to prepare a template.
There are many templates to choose from and very many ways to customize them:


Templates
Create
Email
Raw HTML
My templates
Gallery
Search for templates
Tags
Cyber Monday Journey
Black Friday - Red, White, Black
Cyber Monday - Cyber Sale
Sharp-dressed Father's Day
Happy Father's day
Easter - yellow theme
Easter - clothing store dark the...
Happy Easter - hand drawn ab...

Now that you have a template ready, it is time to create the campaign:





Create new campaign

Regular

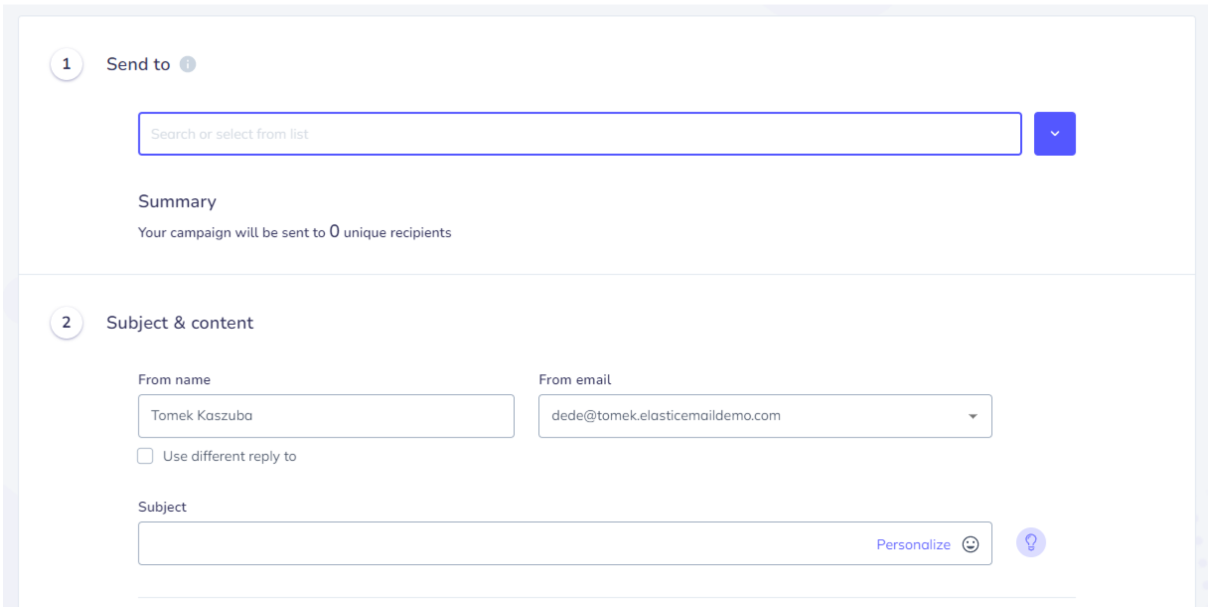
Create and send a single template to your recipients.

Create

A/B test

Send a few variants of templates to see which one will perform better.

Create



1 Send to

Search or select from list

Summary

Your campaign will be sent to 0 unique recipients

2 Subject & content

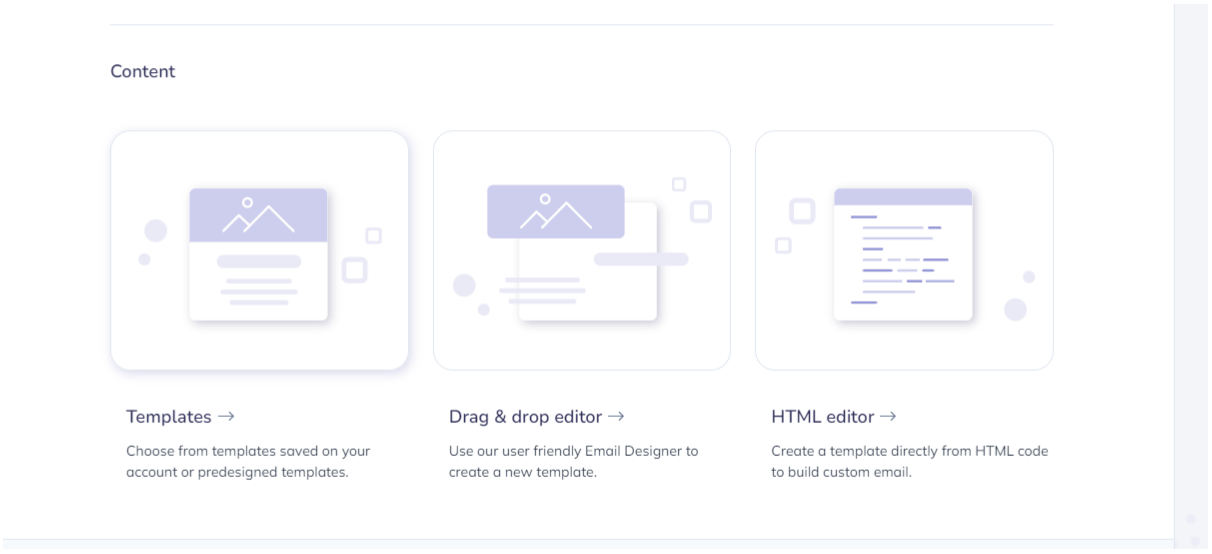
From name: Tomek Kaszuba

From email: dede@tomek.elasticemaildemo.com

Use different reply to

Subject

Personalize


Content
Templates →
Choose from templates saved on your account or predesigned templates.
Drag & drop editor →
Use our user friendly Email Designer to create a new template.
HTML editor →
Create a template directly from HTML code to build custom email.

Try not to get distracted by all the other bells and whistles as this step is easy. In the Campaign Creator just choose the list of contacts, pick a template, and press "Send". The campaign will submit for processing.

How to send to a list or segment

There are few ways to choose a specific contact list or segment you want to send your campaign to. All of them are very simple.

1. If you're sending via Labnify's campaign editor, just select the list or segment from the "SEND TO" box. You can select multiple or just send to "All Contacts".

2. If you're sending via API using the "Email_Send" method, you can select the list or segment with a "lists" or "segments" call. You can select multiple by separating them with a comma or semicolon. If you want to send to "All Contacts", use "0".

Important! Existing list's or segment names should NOT have any spaces in them.

3. If you're sending via SMTP and would like to email your list or segment, you can do do so in one of the following ways:

mylist@lists.labnify.com
or
mysegment@segments.labnify.com

Important! Existing lists or segment names should NOT have any spaces in them.

Important! This list-based email will only work if you submit the request to our SMTP server using your SMTP credentials. So your software (eg. Outlook) would need to be configured with Labnify as your outbound SMTP server.
All of these methods will queue your email for every recipient in that list or segment.

How to avoid the spam folder

Here at Labnify, our team of delivery and customer success experts are available 24/7 to help you solve delivery issues, understand feature functionality and answer any other account questions that you need help with.

Of course, one of those common questions is about why mail is going to the spam folder. This can happen for a wide variety of reasons. This blog post will help you understand what those reasons are and how you can take steps to manage your lists, sender reputation and more in order to get your mail heading to the inbox instead.

Step 1: Setting up your Account for Best Delivery
Before you send your first campaign you need to make sure you're giving yourself the best opportunity for delivery to the inbox. Make sure you've done the following:

-Reviewed our acceptable use policy.
-Verified your sender domain.
-Filled out your profile details accurately and fully.
-Ensure that all mail (including transactional mail) has a clear unsubscribe option and physical address included.

Step 2: Help us Help You

Here are a couple of extra hints when dealing with spam folder issues:

If you've sent less than a 100 emails, and worse if they are 'test' emails (blank, not formatted well etc) - this isn't going to give you or us an accurate picture of what the (if there even is one) issue may be. You need to send a small but representative campaign of mail that is formatted as you intend it to be and to members of your active list before it can be evaluated for potential delivery issues.

Any purchased, borrowed, third party, generic 'industry' lists, scraped addresses or any other contact lists that include addresses that have not directly opted in and asked to receive your mail are against our sending terms and additionally may not comply with laws like the Can-Spam Act or the GDPR. Not only is your mail likely to end up in the spam folder, but your account could get placed under review and all mail stopped completely.
If you decide to send a huge campaign to a single domain (ie Gmail, qq.com, Yahoo etc) or you're sending to an old list (anything over 3-6 months old and your contacts have probably forgotten about signing up to get your mail) and you've not warmed up your verified sender domain then your mail is probably going to have delivery issues.

This can and likely will include your mail going to the spam folder. Use a service to verify your list, then consider sending a confirmation of consent campaign followed by a gradual increase in bulk campaign size.

More actions you can take to reach the inbox

1. If you do send a few test emails (completely formatted as you intend to send them during your campaign) and they land in spam - mark them as not spam - this helps the filters learn that the mail is wanted.
2. Personalize the email you're sending by using the recipient's name. Merge fields can help you with this. Spam filters like to know you're acquainted with the person you're sending to.
3. Ask your recipients to add you to their contact lists or safe sender lists. You can easily ask your recipients to do this when they sign up to receive your mail.
4. Take care of the IPs you're using. Whether you've opted for a private IP or are using one of the shared IP pools it's important that you abide by our terms and ensure that you're using good sending and list management practices.
5. Use a double opt-in list and make sure your subscribers are familiar with your mail and brand. Your contacts should have agreed to receive mail from you and should be expecting it.
6. Your email needs to look good. Spam filters will shut down a campaign that looks like junk mail. It should be balanced between images and text. Find out more about how to make a great email.
7. Keep branding consistent between your social platforms and your email marketing.
8. Make sure any code you're using is clean. Any sloppy code or extra tags will likely cause a problem. For example, copy and pasting HTML from a program like Word can cause format issues and trigger spam filters, we don't recommend it.
9. Use A/X testing to help determine how your campaigns can be improved.
10. Try testing your mail with mail-tester.com to help you determine what might flag a spam filter.
11. Simply avoid "spammy" content. Using words or phrases similar to "click here", "dear recipient", "cash", "make money", "free" can increase the likelihood that your mail ends up being filtered.
12. Use links that work and are connected to legitimate sites. Domains should not be blacklisted. All links that are connected to your email campaigns should be reputable.
13. Avoid link shorteners. There should not be multiple re-directing links.
14. Avoid any attempt at deception. Subject lines that include random characters or start with "Re:" or "Fwd:" or make any deceptive claims will damage your sender reputation and send your mail to the spam folder. Don't use CAPITALS.
15. Do not use random 'From' addresses like cskhgi7779@yourdomain.com for example. We suggest more commonly used ones like 'newsletter@yourdomain.com, support@yourdomain.com, contact@yourdomain.com'. In addition, your mail will not even get delivered if you're not complying with DMARC. You cannot use common addresses from domains like Gmail, Yahoo, Comcast, AOL, Hotmail and more as your from address.
16. Don't send a whole bunch of test emails to a single address. Multiple tests to a single email address may mean that your mail ends up in the spam folder because it looks like you're spamming. Spread your tests out to several addresses at different ISPs.
17. Monitor your mail and choose how often and when you send with care. Sending an email to the same addresses every day might be too often, but sending mail only once every three months might not be enough because your recipients can forget

about you. Determine a consistent sending schedule that will keep your subscribers engaged.

18. Don't send test emails that literally include the word "test" and then are empty otherwise. This is not going to give you any legitimate feedback about how your actual campaigns will perform.
19. Include a text version of your mail. You can find out how to do this in your account settings.
20. Avoid adding attachments if possible, these can be difficult to get through spam filters.
21. Do not use a single, large image in your mail - this looks like spam because it often is. Simple spammy ad mail often utilizes this format - if you don't want to be associated with spam like characteristics, it's time to get a bit more thoughtful and creative when designing your messages.
22. If you take these tips into account and make changes to your mail, lists and sender practices then you'll have a much better chance of reaching the inbox.

API Exclusive product

Let us guide you how to start sending emails via our API or SMTP, supporting both HTTP-API and SMTP-API.

How to send emails via API

Learn how you can send emails with Labnify API.

To use the send command please use POST to https://relay.labnify.com/v4/emails/send with the parameters listed below.

PARAMETERS:
* apikey=your api key,
* subject=email subject,
* from=from email address,
* to=List of email recipients (each email is treated separately, like a BCC). Separated by comma or semicolon. We suggest using the "msgTo" parameter.
* bodyHtml=html email body [optional],
* bodyText=text email body [optional],
* isTransactional - True, if email is transactional (non-bulk, non-marketing, non-commercial). Otherwise, false

Full list of additional parameters is available in our API Documentation.

C#

```
using System;
using System.Collections.Specialized;
using System.Net;
using System.Text;

namespace LabnifyClient
{
  class Program
  {
    static void Main(string[] args)
    {
      NameValueCollection values = new NameValueCollection();
      values.Add("apikey", "00000000-0000-0000-0000-000000000000");
      values.Add("from", "youremail@yourdomain.com");
      values.Add("fromName", "Your Company Name");
      values.Add("to", "recipient1@gmail.com;recipient2@gmail.com");
      values.Add("subject", "Your Subject");
      values.Add("bodyText", "Text Body");
      values.Add("bodyHtml", "<h1>Html Body</h1>");
                    values.Add("isTransactional", true);

      string address = "https://relay.labnify.com/v4/emails/send";

      string response = Send(address, values);

      Console.WriteLine(response);
    }

    static string Send(string address, NameValueCollection values)
    {
      using (WebClient client = new WebClient())
      {
        try
        {
          byte[] apiResponse = client.UploadValues(address, values);
          return Encoding.UTF8.GetString(apiResponse);

        }
        catch (Exception ex)
        {
          return "Exception caught: " + ex.Message + "\n" + ex.StackTrace;
        }
      }
    }
```

```
   }
}
```

Java

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.URL;
import java.net.URLConnection;
import java.net.URLEncoder;

public class LabnifyClient {

 public static String Send(String userName, String apiKey, String from, String fromName,
String subject, String body, String to, String isTransactional) {

 try {

 String encoding = "UTF-8";

 String data = "apikey=" + URLEncoder.encode(apiKey, encoding);
 data += "&from=" + URLEncoder.encode(from, encoding);
 data += "&fromName=" + URLEncoder.encode(fromName, encoding);
 data += "&subject=" + URLEncoder.encode(subject, encoding);
 data += "&bodyHtml=" + URLEncoder.encode(body, encoding);
 data += "&to=" + URLEncoder.encode(to, encoding);
 data += "&isTransactional=" + URLEncoder.encode(isTransactional, encoding);

 URL url = new URL("https://relay.labnify.com/v4/emails/send");
 URLConnection conn = url.openConnection();
 conn.setDoOutput(true);
 OutputStreamWriter wr = new OutputStreamWriter(conn.getOutputStream());
 wr.write(data);
 wr.flush();
 BufferedReader rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));
 String result = rd.readLine();
 wr.close();
 rd.close();

 return result;
 }

 catch(Exception e) {

 e.printStackTrace();
 }
 }

}
```

PHP

```
<?php
$url = 'https://relay.labnify.com/v4/emails/send';

try{
        $post = array('from' => 'youremail@yourdomain.com',
                      'fromName' => 'Your Company Name',
                      'apikey' => '00000000-0000-0000-0000-000000000000',
                      'subject' => 'Your Subject',
                      'to' => 'recipient1@gmail.com;recipient2@gmail.com',
                      'bodyHtml' => '<h1>Html Body</h1>',
                      'bodyText' => 'Text Body',
                      'isTransactional' => false);

                      $ch = curl_init();
                      curl_setopt_array($ch, array(
            CURLOPT_URL => $url,
                                   CURLOPT_POST => true,
                                   CURLOPT_POSTFIELDS => $post,
            CURLOPT_RETURNTRANSFER => true,
            CURLOPT_HEADER => false,
                                   CURLOPT_SSL_VERIFYPEER => false
        ));

        $result=curl_exec ($ch);
        curl_close ($ch);

        echo $result;
}
catch(Exception $ex){
            echo $ex->getMessage();
}
?>
```

Python

```
import requests
import json
from enum import Enum
```

```
class ApiClient:
	apiUri = 'https://relay.labnify.com/v4'
	apiKey = '00000000-0000-0000-0000-0000000000000'

	def Request(method, url, data):
		data['apikey'] = ApiClient.apiKey
		if method == 'POST':
			result = requests.post(ApiClient.apiUri + url, data = data)
		elif method == 'PUT':
			result = requests.put(ApiClient.apiUri + url, data = data)
		elif method == 'GET':
			attach = ''
			for key in data:
				attach = attach + key + '=' + data[key] + '&'
			url = url + '?' + attach[:-1]
			result = requests.get(ApiClient.apiUri + url)

		jsonMy = result.json()

		if jsonMy['success'] is False:
			return jsonMy['error']

		return jsonMy['data']

def Send(subject, EEfrom, fromName, to, bodyHtml, bodyText, isTransactional):
	return ApiClient.Request('POST', '/email/send', {
		'subject': subject,
		'from': EEfrom,
		'fromName': fromName,
		'to': to,
		'bodyHtml': bodyHtml,
		'bodyText': bodyText,
		'isTransactional': isTransactional})

print(Send("Your Subject", "youremail@yourdomain.com", "Your Company Name",
"recipient1@gmail.com;recipient2@gmail.com", "<h1>Html Body</h1>", "Text Body", True))
```

License

Sending email with attachments via API

Learn how to send email with attachments via Labnify API.

Sending with attachments is not that much different from Mail Merge.
You need to provide your attachments as the Multipart/Form-Data POST field in your Send request, so the specific request's parameter for attachments does not exist.
There is no limit on how many attachments you can send with the email.

Note: There is currently a limit of 10 MB/overall email size.

C#

```
using System;
using System.Collections.Specialized;
using System.IO;
using System.Net;
using System.Net.Http;
using System.Text;
using System.Threading;

namespace LabnifyClient
{
    class Program
    {
        static void Main(string[] args)
        {
            NameValueCollection values = new NameValueCollection();
            values.Add("apikey", "00000000-0000-0000-0000-000000000000");
            values.Add("from", "youremail@yourdomain.com");
            values.Add("fromName", "Your Company Name");
            values.Add("subject", "Your Subject");
            values.Add("bodyText", "Text Body");
            values.Add("to", "recipient1@gmail.com;recipient2@gmail.com");
            values.Add("bodyHtml", "&lt;h1&gt;Html Body&lt;/h1&gt;");

            var filepath = "C:\\example\\helloWorld.txt";
            var file = File.OpenRead(filepath);

            var filesStream = new Stream[] { file };
            var filenames = new string[] { "filenameForInbox.txt" };
            var URL = "https://relay.labnify.com/v4/emails/send";

            string result = Upload(URL, values, filesStream, filenames);

            Console.WriteLine(result);
        }
        public static string Upload(string actionUrl, NameValueCollection values, Stream[]
paramFileStream = null, string[] filenames = null)
        {
            using (var client = new HttpClient())
            using (var formData = new MultipartFormDataContent())
            {
                foreach (string key in values)
                {
                    HttpContent stringContent = new StringContent(values[key]);
                    formData.Add(stringContent, key);
                }

                for (int i = 0; i < paramFileStream.Length; i++)
                {
                    HttpContent fileStreamContent = new StreamContent(paramFileStream[i]);
                    formData.Add(fileStreamContent, "file" + i, filenames[i]);
                }

                var response = client.PostAsync(actionUrl, formData).Result;
                if (!response.IsSuccessStatusCode)
                {
                    throw new Exception(response.Content.ReadAsStringAsync().Result);
                }

                return response.Content.ReadAsStringAsync().Result;
            }
        }

    }
}
```

PHP

```
<?php
$url = 'https://relay.labnify.com/v4/emails/send';
$filename = "helloWorld.txt";
$file_name_with_full_path = realpath('./'.$filename);
$filetype = "text/plain"; // Change correspondingly to the file type

try{
        $post = array('from' => 'youremail@yourdomain.com',
                'fromName' => 'Your Company Name',
                'apikey' => '00000000-0000-0000-0000-000000000000',
                'subject' => 'Your Subject',
                'bodyHtml' => '&lt;h1&gt;Html Body&lt;/h1&gt;',
                'bodyText' => 'Text Body',
                'to' => 'recipient1@gmail.com;recipient2@gmail.com',
                'isTransactional' => false,
                'file_1' => new CurlFile($file_name_with_full_path, $filetype, $filename));

        $ch = curl_init();

        curl_setopt_array($ch, array(
          CURLOPT_URL => $url,
          CURLOPT_POST => true,
          CURLOPT_POSTFIELDS => $post,
          CURLOPT_RETURNTRANSFER => true,
          CURLOPT_HEADER => false,
          CURLOPT_SSL_VERIFYPEER => false
        ));

        $result=curl_exec ($ch);
        curl_close ($ch);

        echo $result;
}
catch(Exception $ex){
  echo $ex->getMessage();
}
?>
```

Java

```
--------------LabnifyClient.java--------
package com.labnify.app;

import java.io.IOException;
import java.util.HashMap;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

public class LabnifyClient {
  public static void main(String[] args) {
    API client = new API();
    ArrayList<FileData>files = new ArrayList<>();
    String filename = "helloWorld.txt";

    Path path = Paths.get("C:\\example\\" + filename);
    byte[] data = null;
    try {
        data = Files.readAllBytes(path);
    } catch (IOException ex) {
        Logger.getLogger(LabnifyClient.class.getName()).log(Level.SEVERE, null, ex);
    }

    FileData contactsFile = new FileData();
    contactsFile.contentType = "text/plain"; // Change correspondingly to the file type
    contactsFile.fileName = filename;
    contactsFile.content = data;
    files.add(contactsFile);

    HashMap<String, String> values = new HashMap<>();
    values.put("apikey", "00000000-0000-0000-0000-000000000000");
    values.put("from", "youremail@yourdomain.com");
    values.put("fromName", "Your Company Name");
    values.put("subject", "Your Subject");
    values.put("to", "recipient1@gmail.com;recipient2@gmail.com");
    values.put("bodyText", "Text Body");
    values.put("bodyHtml", "&lt;h1&gt;Html Body&lt;/h1&gt;");

    try {
        String result = client.httpPostFile("/email/send", files, values);
        System.out.println(result);
    } catch (Exception ex) {
        Logger.getLogger(LabnifyClient.class.getName()).log(Level.SEVERE, null, ex);
    }
  }
}


-------------- API.java ----------------
package com.labnify.app;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.nio.charset.Charset;
import java.util.Map;

public class API {
   public static String API_KEY = "";
   protected static String API_URI = "https://relay.labnify.com/v4";

   protected String httpPostFile(String targetURL, Iterable<FileData> fileData, Map<String,
String> values) throws Exception {
      if (targetURL == null) throw new IllegalArgumentException("targetURL");
      if (values == null) throw new IllegalArgumentException("values");
      if (fileData == null) throw new IllegalArgumentException("fileData");

      HttpURLConnection connection = null;
      URL url = null;
      String urlParameters = null;
      String urlParametersLength = null;

      try {
         url = new URL(API_URI + targetURL);
         urlParameters = loadUrlParameters(values);
         urlParametersLength = Integer.toString(urlParameters.getBytes().length);
         String boundary = String.valueOf(System.currentTimeMillis());
         byte[] boundarybytes = ("\r\n--" + boundary +
"\r\n").getBytes(Charset.forName("ASCII"));

         connection = (HttpURLConnection)url.openConnection();
         connection.setRequestProperty("Content-Type", "multipart/form-data; boundary=" +
boundary);
         connection.setRequestMethod("POST");
         connection.setRequestProperty("Connection", "Keep-Alive");
         connection.setRequestProperty("Content-Length", "" + urlParametersLength);
         connection.setUseCaches(false);
         connection.setDoInput(true);
         connection.setDoOutput(true);

         //Send request
         DataOutputStream wr = new DataOutputStream(connection.getOutputStream ());

      String formdataTemplate = "Content-Disposition: form-data; name=\"%s\"\r\n\r\n%s";
      for (String key : values.keySet())
      {
         wr.write(boundarybytes, 0, boundarybytes.length);
         String formitem = String.format(formdataTemplate, key, values.get(key));
         byte[] formitembytes = formitem.getBytes(Charset.forName("UTF8"));
         wr.write(formitembytes, 0, formitembytes.length);
      }

      if(fileData != null){
         for(FileData file : fileData){
            wr.write(boundarybytes, 0, boundarybytes.length);
            String headerTemplate = "Content-Disposition: form-data;
name=\"filefoobarname\"; filename=\"%s\"\r\nContent-Type: %s\r\n\r\n";
            String header = String.format(headerTemplate, file.fileName, file.contentType);
            byte[] headerbytes = header.getBytes(Charset.forName("UTF8"));
            wr.write(headerbytes, 0, headerbytes.length);
            wr.write(file.content, 0, file.content.length);
          }
        }

        byte[] trailer = ("\r\n--" + boundary + "--\r\n").getBytes(Charset.forName("ASCII"));
        wr.write(trailer, 0, trailer.length);
        wr.flush ();
        wr.close ();

        //Get Response
        InputStream is = connection.getInputStream();
        BufferedReader rd = new BufferedReader(new InputStreamReader(is));
        String line;
        StringBuilder response = new StringBuilder();
        while((line = rd.readLine()) != null) {
          response.append(line);
          response.append('\r');
        }
        rd.close();

        return response.toString();

        } catch (IOException e) {
          e.printStackTrace();
          return null;

        } finally {
          if(connection != null) {
            connection.disconnect();
          }
        }
      }

      private String loadUrlParameters(Map<String, String> values) {
        StringBuilder sb = new StringBuilder();

        values.keySet().forEach((key) -> {
          if (sb.length() > 0) {
            sb.append("&");
          }
          String value = values.get(key);
          try {
                sb.append((key != null ? URLEncoder.encode(key, "UTF-8") : ""));
                sb.append("=");
                sb.append(value != null ? URLEncoder.encode(value, "UTF-8") : "");
            } catch (UnsupportedEncodingException e) {
                throw new RuntimeException("This method is not supported", e);
            }
        });

        return sb.toString();
    }
}


---------------- FileData.java -----------------------
package com.labnify.app;

import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

public class FileData {
    public byte[] content;

    public String contentType;

    public String fileName;

    public void ReadFrom(String pathWithFileName) throws Exception
    {
        Path path = Paths.get(pathWithFileName);
        content = Files.readAllBytes(path);
        fileName = path.getFileName().toString();
        contentType = null;
    }

    public static FileData CreateFromFile(String pathWithFileName) throws Exception
    {
        FileData fileData = new FileData();
        fileData.ReadFrom(pathWithFileName);
        return fileData;
    }
}
```

Python

```
import requests
import json
from enum import Enum

class ApiClient:
  apiUri = 'https://relay.labnify.com/v4'
  apiKey = '00000000-0000-0000-0000-000000000000'

  def Request(method, url, data, attachs=None):
    data['apikey'] = ApiClient.apiKey
    if method == 'POST':
      result = requests.post(ApiClient.apiUri + url, params = data, files = attachs)
    elif method == 'PUT':
      result = requests.put(ApiClient.apiUri + url, params = data)
    elif method == 'GET':
      attach = ''
      for key in data:
        attach = attach + key + '=' + data[key] + '&'
      url = url + '?' + attach[:-1]
      result = requests.get(ApiClient.apiUri + url)

    jsonMy = result.json()

    if jsonMy['success'] is False:
      return jsonMy['error']

    return jsonMy['data']

def Send(subject, EEfrom, fromName, to, bodyHtml, bodyText, isTransactional,
attachmentFiles = []):
  tmp_attachments = []
  for name in attachmentFiles:
    tmp_attachments.append(('attachments', open(name, 'rb')))

  return ApiClient.Request('POST', '/email/send', {
        'subject': subject,
        'from': EEfrom,
        'fromName': fromName,
        'to': to,
        'bodyHtml': bodyHtml,
        'bodyText': bodyText,
        'isTransactional': isTransactional}, tmp_attachments)

attachments = []
attachments.append('C:/example/helloWorld.txt')
```

print(Send("Your Subject", "youremail@yourdomain.com", "Your Company Name", "recipient1@gmail.com;recipient2@gmail.com", "&lt;h1&gt;Html Body&lt;/h1&gt;", "Text Body", True, attachments))

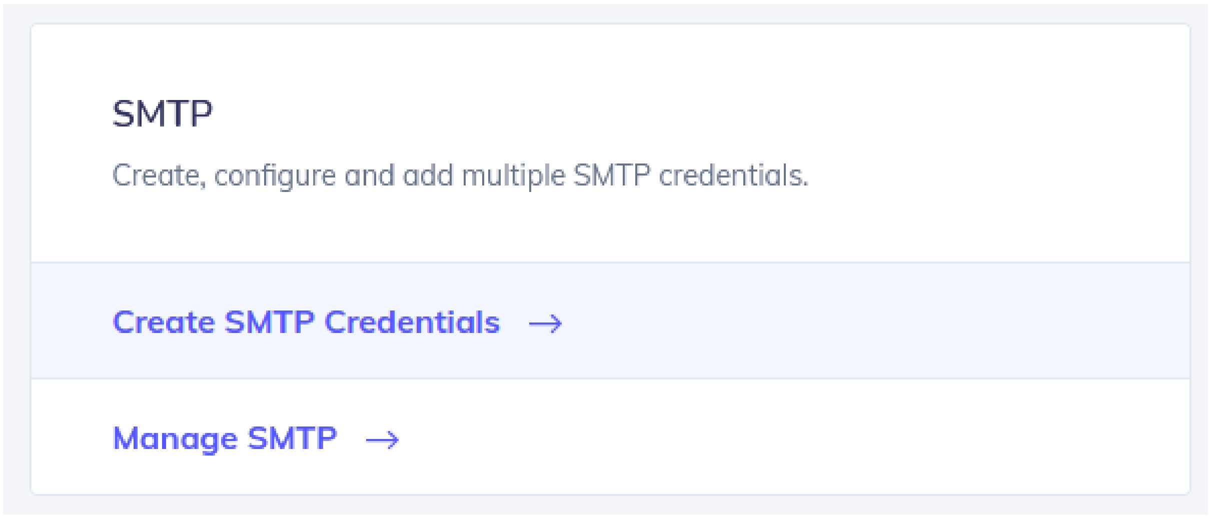
License
All code samples are licensed under MIT license.

Creating your SMTP Credentials

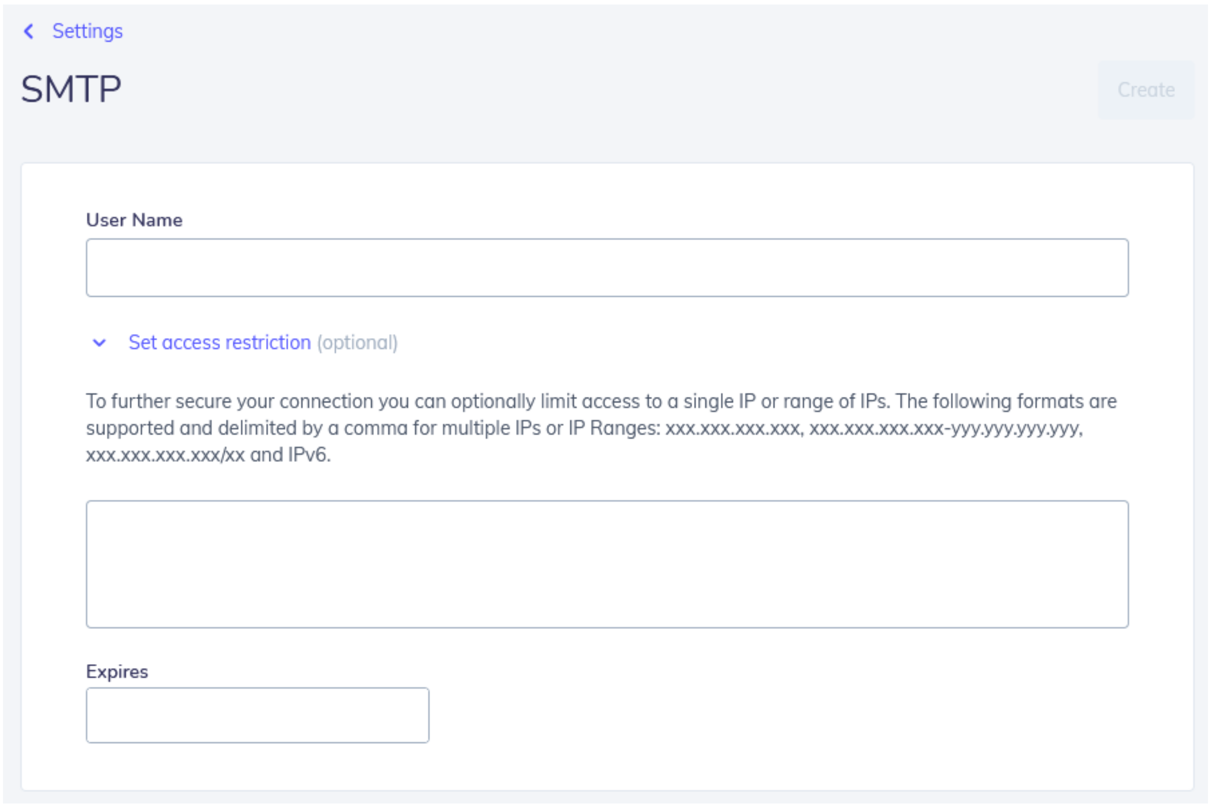
Create and manage your SMTP Credentials

Note: Each account can store up to 15 unique SMTP Credentials

Almost all third party SMTP clients, Servers, CMSs and Plugins will use these settings to connect and relay mail through your Labnify Account. We don't know of any software that supports SMTP connections that do not work with Labnify but there is a list of CMSs and Plugins that are designed to work specifically with our platform.


SMTP
Create, configure and add multiple SMTP credentials.
Create SMTP Credentials →
Manage SMTP →

Create SMTP Credentials
New accounts do not have any SMTP Credentials. Click this button to create a new one.



To further secure your connection you can optionally limit access to a single IP or range of IPs. The following formats are supported and delimited by a comma for multiple IPs or IP Ranges: xxx.xxx.xxx.xxx, xxx.xxx.xxx.xxx-yyy.yyy.yyy.yyy, xxx.xxx.xxx.xxx/xx and IPv6.

Username

Your Labnify Account SMTP Username defaults to your Labnify Account email address that you use to log in. You can use another email address, but it has to be unique across the entire system - just as with account email addresses. Here you can also set IP access restriction.
After clicking the Create button, a pop up will appear with your newly created SMTP Credential details.

Password
A unique password for your SMTP connection. Copy it and save it in a safe place. **Once you close this window you will not be able to retrieve it.** After this step, our system will obfuscate it except for the last 5 digits for your security reasons.

Server You can either use **smtp.labnify.com** or **smtp25.labnify.com**

Port We support the following connection ports:
25, 2525, 587, 465
We support TLS 1.2 and SSL connections. Make sure the connection port you are using is open on your own server.

Using Postfix? If you are configuring postfix as a local relay, make sure you include the following three lines in your postfix configuration
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = static:yourusername:your api key

smtp_sasl_security_options = noanonymous
You can also **Edit, Change** or **Add other SMTP Credentials**.

SMTP connection limits
20 concurrent requests from single IP
Single command timeout is 2 minutes
Multiple emails can be submitted in one session.

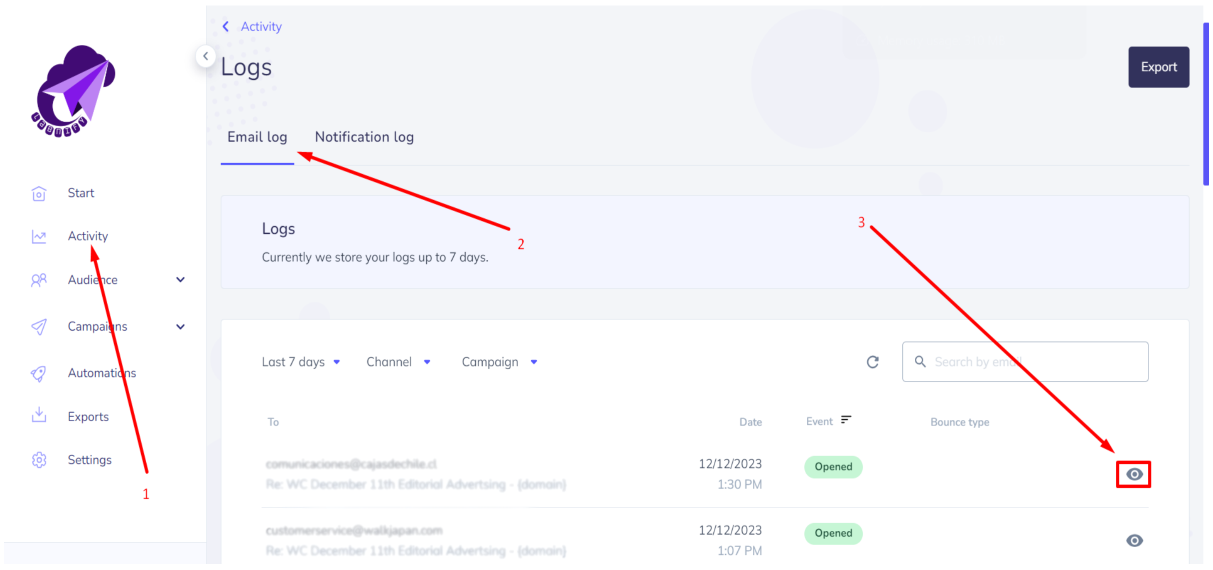
How to find a message ID

Labnify uses MessageID as a way to uniquely identify emails. If you have a question or concern about a specific email, Labnify Customer Success team can use the MessageID to quickly look up details regarding your email.

There are a few ways you can locate an email's MessageID.

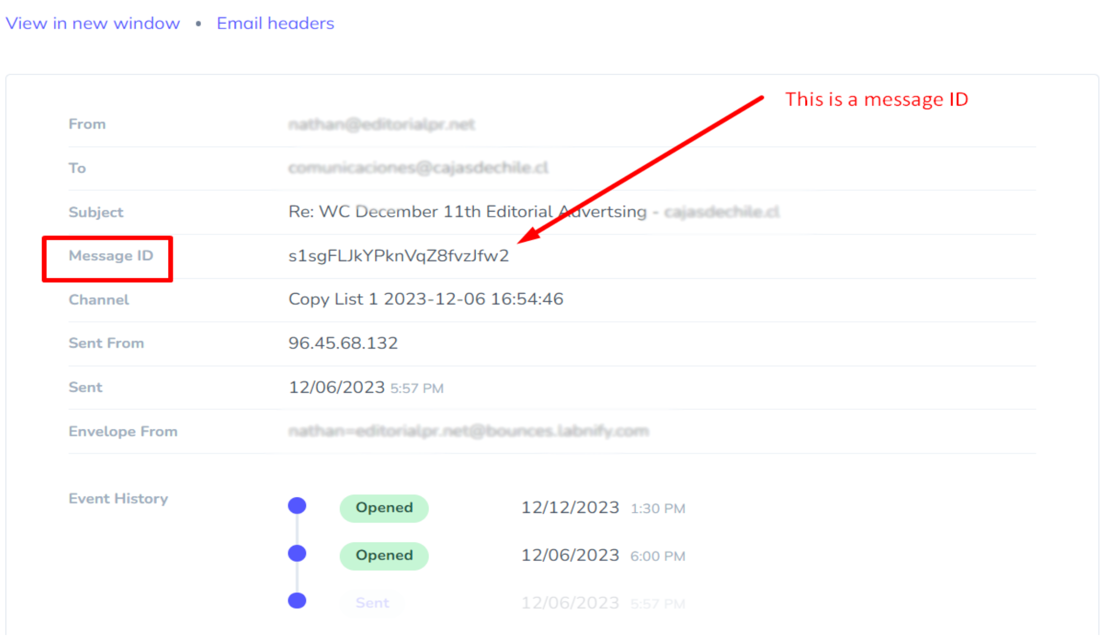
From Your Email Logs

Go to the Activity Screen>View>Email log and click the "eye" icon besides an email. A popup will appear with email's details, including the MessageID:



Logs
Currently we store your logs up to 7 days.

To	Date	Event	Bounce type
comunicaciones@cajasdechile.cl Re: WC December 11th Editorial Advertsing - {domain}	12/12/2023 1:30 PM	Opened	
customerservice@walkjapan.com Re: WC December 11th Editorial Advertsing - {domain}	12/12/2023 1:07 PM	Opened	


View in new window • Email headers

This is a message ID

From	[redacted]
To	[redacted]
Subject	Re: WC December 11th Editorial Advertsing - [redacted]
Message ID	s1sgFLJkYPknVqZ8fvzJfw2
Channel	Copy List 1 2023-12-06 16:54:46
Sent From	96.45.68.132
Sent	12/06/2023 5:57 PM
Envelope From	[redacted]

Event History		
●	Opened	12/12/2023 1:30 PM
●	Opened	12/06/2023 6:00 PM
●	Sent	12/06/2023 5:57 PM

MessageIDs are available in your Email Logs. Within the UI, you just click the "eye" icon on the email. The Email Viewer will display with details about the email, including the MessageID. The same details are available in the exported Email Logs.

From Your Email Client

The MessageID is built into the email as a "header". You can find the email's MessageID right from within your email client by viewing what is often called the "original" or "source".

Google has a help document that shows you how to find this option in most major email clients.

Using mail merge function

Learn how to send customized emails to multiple recipients with Mail Merge function.

Using this technique you can define an email template which contains special fields such as {firstname}, {lastname} or many others and send personalized emails in a very efficient way to your clients.

For example: *Dear {firstname} {lastname},*
This is our monthly newsletter. We hope you have been enjoying our service. You can access your account at {link}.

If you would like to allow custom headers, please make sure this has been activated on your advanced options screen.

Create a CSV file

Create a CSV file like the following:
"ToEmail","FirstName","LastName"
"email1@email.com","Joe","Plumber"
"email2@email.com", "Jackie","Smith"

The first column of the data must be ToEmail and will be used as the recipients of the merged email. The other columns represent the data fields that you can use in your email. You can have as many columns as you need. If you have the header FirstName you can include that in your email template with {FirstName} and all references to that field will be replaced before sending.
Please note that when you use this method, recipients in the "**To**" parameter will be ignored.

Send Email

To use the send POST command to https://relay.labnify.com/v4/emails/send with the following form values:
username=your account email address,
api_key=your API key,
from=from email address,
from_name=display name for from email address,
subject=email subject,
body_html=html email body [optional],
body_text=text email body [optional],
mergesourcefilename=your CSV file name sent with this call as an attachment,
charset=text value of encoding for example: iso-8859-1, windows-1251, utf-8, us-ascii, windows-1250 and more…,
encoding for None, 1 for Raw7Bit, 2 for Raw8Bit, 3 for QuotedPrintable, 4 for Base64 (Default), 5 for Uue (**note that you can also provide the text version such as "Raw7Bit" for value 1)**.

If sent correctly you will receive a response like:
{"success":true,"data":{"transactionid":"d6126dc1-d012-4d4c-b75b-43ak8dd7df9c","messageid":"7jAW8_jz5Iaw7DjsPaaH9A2"}}'

This contains the transaction ID of your send job. You can use this transaction ID to check on the statistics of the given job using the Get Status API.

Your email will be sent to each email address in the CSV file and the data will be merged with the CSV data.

Note: There is a limit of 100000 emails / uploaded CSV file for larger jobs please split into separate calls.

Mail Merge with single recipient

You can also use the merge fields when sending to a single recipient. You can, of course, create a CSV file with one recipient and follow our instructions mentioned above, or you could create the merge fields' values on the go:
Simply use a merge field in your bodyHtml or bodyText ("Hello, {firstname}" etc.) and provide its value at the same request by following the given convention:
merge_X = Y
For the given firstname, it would be:
merge_firstname=John
Implement how many fields you need and provide them with the Send request just as you would provide a subject or body HTML, and you are ready to go!

C#

```
using System;
using System.Collections.Specialized;
using System.IO;
using System.Net;
using System.Net.Http;
using System.Text;
using System.Threading;

namespace LabnifyClient
{
   class Program
   {
      static void Main(string[] args)
      {
         NameValueCollection values = new NameValueCollection();
         values.Add("apikey", "00000000-0000-0000-0000-000000000000");
         values.Add("from", "youremail@yourdomain.com");
         values.Add("fromName", "Your Company Name");
         values.Add("subject", "Your Subject");
         values.Add("bodyText", "Hello, {firstname}");
         values.Add("bodyHtml", "<h1>Hello, {firstname}</h1>");
         values.Add("mergesourcefilename", "mycontacts.csv"); // Same as the name of the
file in the 'filenames' array

         var filepath = "C:\\example\\contacts.csv";
         var file = File.OpenRead(filepath);

         // multiple files can be sent using this method
         var filesStream = new Stream[] { file };
         var filenames = new string[] { "mycontacts.csv" };
         var URL = "https://relay.labnify.com/v4/emails/send";

         string result = Upload(URL, values, filesStream, filenames);
```

```
            Console.WriteLine(result);
        }

        public static string Upload(string actionUrl, NameValueCollection values, Stream[]
paramFileStream = null, string[] filenames = null)
        {
            using (var client = new HttpClient())
            using (var formData = new MultipartFormDataContent())
            {
                foreach (string key in values)
                {
                    HttpContent stringContent = new StringContent(values[key]);
                    formData.Add(stringContent, key);
                }

                for (int i = 0; i < paramFileStream.Length; i++)
                {
                    HttpContent fileStreamContent = new StreamContent(paramFileStream[i]);
                    formData.Add(fileStreamContent, "file" + i, filenames[i]);
                }

                var response = client.PostAsync(actionUrl, formData).Result;
                if (!response.IsSuccessStatusCode)
                {
                    throw new Exception(response.Content.ReadAsStringAsync().Result);
                }

                return response.Content.ReadAsStringAsync().Result;
            }
        }

    }
}
```

PHP

```
<?php
$url = 'https://relay.labnify.com/v4/emails/send';
$filename = "contacts.csv";
$file_name_with_full_path = realpath('./'.$filename);

try{
            $post = array('from' => 'youremail@yourdomain.com',
                        'fromName' => 'Your Company Name',
                        'apikey' => '00000000-0000-0000-0000-000000000000',
                        'subject' => 'Your Subject',
                        'bodyHtml' => '<h1>Hello, {firstname}</h1>',
                        'bodyText' => 'Hello, {firstname}',
                        'isTransactional' => false,
                        'file_contacts' => new CurlFile($file_name_with_full_path, 'text/csv',
$filename),
                        'mergesourcefilename' => $filename);

                        $ch = curl_init();

                        curl_setopt_array($ch, array(
                                  CURLOPT_URL => $url,
                                  CURLOPT_POST => true,
                                  CURLOPT_POSTFIELDS => $post,
                                  CURLOPT_RETURNTRANSFER => true,
                                  CURLOPT_HEADER => false,
                                  CURLOPT_SSL_VERIFYPEER => false
                        ));

                        $result=curl_exec ($ch);

                if ($result === FALSE)
                   { echo curl_error($ch); }
                else
                   { echo $result; }

                        curl_close ($ch);
}
catch(Exception $ex){
            echo $ex->getMessage();
}
?>
```

JAVA

```
--------------LabnifyClient.java--------
package com.labnify.app;

import java.io.IOException;
import java.util.HashMap;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
```

```
public class LabnifyClient {
   public static void main(String[] args) {
      API client = new API();
      ArrayListfiles = new ArrayList<>();
      String filename = "Contacts.csv";

      Path path = Paths.get("C:\\example\\" + filename);
      byte[] data = null;
      try {
         data = Files.readAllBytes(path);
      } catch (IOException ex) {
         Logger.getLogger(LabnifyClient.class.getName()).log(Level.SEVERE, null, ex);
      }

      FileData contactsFile = new FileData();
      contactsFile.contentType = "text/csv";
      contactsFile.fileName = filename;
      contactsFile.content = data;
      files.add(contactsFile);

      HashMap<String, String> values = new HashMap<>();
      values.put("apikey", "00000000-0000-0000-0000-000000000000");
      values.put("from", "youremail@yourdomain.com");
      values.put("fromName", "Your Company Name");
      values.put("subject", "Your Subject");
      values.put("bodyText", "Hello, {firstname}");
      values.put("bodyHtml", "<h1>Hello, {firstname}</h1>");
      values.put("mergesourcefilename", filename);

      try {
         String result = client.httpPostFile("/email/send", files, values);
         System.out.println(result);
      } catch (Exception ex) {
         Logger.getLogger(LabnifyClient.class.getName()).log(Level.SEVERE, null, ex);
      }
   }
}


-------------- API.java ----------------
package com.labnify.app;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.nio.charset.Charset;
import java.util.Map;

public class API {

  public static String API_KEY = "";
  protected static String API_URI = "https://relay.labnify.com/v4";

  protected String httpPostFile(String targetURL, Iterable fileData, Map<String, String>
values) throws Exception {
    if (targetURL == null) throw new IllegalArgumentException("targetURL");
    if (values == null) throw new IllegalArgumentException("values");
    if (fileData == null) throw new IllegalArgumentException("fileData");

    HttpURLConnection connection = null;
    URL url = null;
    String urlParameters = null;
    String urlParametersLength = null;

    try {
      url = new URL(API_URI + targetURL);
      urlParameters = loadUrlParameters(values);
      urlParametersLength = Integer.toString(urlParameters.getBytes().length);
      String boundary = String.valueOf(System.currentTimeMillis());
      byte[] boundarybytes = ("\r\n--" + boundary +
"\r\n").getBytes(Charset.forName("ASCII"));

      connection = (HttpURLConnection)url.openConnection();
      connection.setRequestProperty("Content-Type", "multipart/form-data; boundary=" +
boundary);
      connection.setRequestMethod("POST");
      connection.setRequestProperty("Connection", "Keep-Alive");
      connection.setRequestProperty("Content-Length", "" + urlParametersLength);
      connection.setUseCaches(false);
      connection.setDoInput(true);
      connection.setDoOutput(true);

      //Send request
      DataOutputStream wr = new DataOutputStream(connection.getOutputStream ());

    String formdataTemplate = "Content-Disposition: form-data; name=\"%s\"\r\n\r\n%s";
    for (String key : values.keySet())
    {
      wr.write(boundarybytes, 0, boundarybytes.length);
      String formitem = String.format(formdataTemplate, key, values.get(key));
      byte[] formitembytes = formitem.getBytes(Charset.forName("UTF8"));
      wr.write(formitembytes, 0, formitembytes.length);
    }

    if(fileData != null){
      for(FileData file : fileData){
          wr.write(boundarybytes, 0, boundarybytes.length);
          String headerTemplate = "Content-Disposition: form-data;
name=\"filefoobarname\"; filename=\"%s\"\r\nContent-Type: %s\r\n\r\n";
          String header = String.format(headerTemplate, file.fileName, file.contentType);
          byte[] headerbytes = header.getBytes(Charset.forName("UTF8"));
          wr.write(headerbytes, 0, headerbytes.length);
          wr.write(file.content, 0, file.content.length);
      }
    }

    byte[] trailer = ("\r\n--" + boundary + "--\r\n").getBytes(Charset.forName("ASCII"));
    wr.write(trailer, 0, trailer.length);
    wr.flush ();
    wr.close ();

    //Get Response
    InputStream is = connection.getInputStream();
    BufferedReader rd = new BufferedReader(new InputStreamReader(is));
    String line;
    StringBuilder response = new StringBuilder();
    while((line = rd.readLine()) != null) {
      response.append(line);
      response.append('\r');
    }
    rd.close();

    return response.toString();

    } catch (IOException e) {
      e.printStackTrace();
      return null;

    } finally {
      if(connection != null) {
          connection.disconnect();
      }
    }
  }
  private String loadUrlParameters(Map<String, String> values) {
    StringBuilder sb = new StringBuilder();

    values.keySet().forEach((key) -> {
      if (sb.length() > 0) {
          sb.append("&");
      }
      String value = values.get(key);
      try {
          sb.append((key != null ? URLEncoder.encode(key, "UTF-8") : ""));
          sb.append("=");
          sb.append(value != null ? URLEncoder.encode(value, "UTF-8") : "");
      } catch (UnsupportedEncodingException e) {
          throw new RuntimeException("This method is not supported", e);
      }
    });

    return sb.toString();
  }
}


---------------- FileData.java -----------------------
package com.labnify.app;

import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

public class FileData {
  public byte[] content;

  public String contentType;

  public String fileName;

  public void ReadFrom(String pathWithFileName) throws Exception
  {
      Path path = Paths.get(pathWithFileName);
    content = Files.readAllBytes(path);
    fileName = path.getFileName().toString();
    contentType = null;
  }

  public static FileData CreateFromFile(String pathWithFileName) throws Exception
  {
    FileData fileData = new FileData();
    fileData.ReadFrom(pathWithFileName);
    return fileData;
  }
}
```

Python

```
import requests
import json
from enum import Enum

class ApiClient:
	apiUri = 'https://relay.labnify.com/v4'
	apiKey = '00000000-0000-0000-0000-000000000000'

	def Request(method, url, data, attachs=None):
		data['apikey'] = ApiClient.apiKey
		if method == 'POST':
			result = requests.post(ApiClient.apiUri + url, params = data, files = 
attachs)
		elif method == 'PUT':
			result = requests.put(ApiClient.apiUri + url, params = data)
		elif method == 'GET':
			attach = ''
			for key in data:
				attach = attach + key + '=' + data[key] + '&'
			url = url + '?' + attach[:-1]
			result = requests.get(ApiClient.apiUri + url)

		jsonMy = result.json()

		if jsonMy['success'] is False:
			return jsonMy['error']

		return jsonMy['data']

def Send(subject, EEfrom, fromName, bodyHtml, bodyText, isTransactional, 
contactsSourceFilename, attachmentFiles = []):
	attachments = []
	for name in attachmentFiles:
		attachments.append(('attachments'.name, open(name, 'rb')))

	return ApiClient.Request('POST', '/email/send', {
		'subject': subject,
		'from': EEfrom,
		'fromName': fromName,
                    'bodyHtml': bodyHtml,
                    'bodyText': bodyText,
                    'mergesourcefilename': contactsSourceFilename,
                    'isTransactional': isTransactional}, attachments)

attachments = []
attachments.append('C:/example/Contacts.csv')
print(Send("Your Subject", "youremail@yourdomain.com", "Your Company Name",
"<h1>Hello, {firstname}</h1>", "Hello, {firstname}", True, "Contacts.csv", attachments))
```

License