Action Mailbox Basics

This guide provides you with all you need to get started in receiving emails to your application.

After reading this guide, you will know:

- How to receive email within a Rails application.
- How to configure Action Mailbox.
- How to generate and route emails to a mailbox.
- How to test incoming emails.

What is Action Mailbox?

Action Mailbox routes incoming emails to controller-like mailboxes for processing in Rails. It ships with ingresses for Mailgun, Mandrill, Postmark, and SendGrid. You can also handle inbound mails directly via the built-in Exim, Postfix, and Qmail ingresses.

The inbound emails are turned into `InboundEmail` records using Active Record and feature lifecycle tracking, storage of the original email on cloud storage via Active Storage, and responsible data handling with on-by-default incineration.

These inbound emails are routed asynchronously using Active Job to one or several dedicated mailboxes, which are capable of interacting directly with the rest of your domain model.

Setup

Install migrations needed for `InboundEmail` and ensure Active Storage is set up:

```
$ bin/rails action_mailbox:install
$ bin/rails db:migrate
```

Configuration

Exim

Tell Action Mailbox to accept emails from an SMTP relay:

```ruby
# config/environments/production.rb
config.action_mailbox.ingress = :relay
```

Generate a strong password that Action Mailbox can use to authenticate requests to the relay ingress.

Use `bin/rails credentials:edit` to add the password to your application's encrypted credentials under `action_mailbox.ingress_password`, where Action Mailbox will automatically find it:

```
action_mailbox:
  ingress_password: ...
```

Alternatively, provide the password in the `RAILS_INBOUND_EMAIL_PASSWORD` environment variable.

Configure Exim to pipe inbound emails to `bin/rails action_mailbox:ingress:exim`, providing the `URL` of the relay ingress and the `INGRESS_PASSWORD` you previously generated. If your application lived at `https://example.com`, the full command would look like this:

```
$ bin/rails action_mailbox:ingress:exim URL=https://example.com/rails/action_mailbox/relay/i
```

Mailgun

Give Action Mailbox your Mailgun Signing key (which you can find under Settings -> Security & Users -> API security in Mailgun), so it can authenticate requests to the Mailgun ingress.

Use `bin/rails credentials:edit` to add your Signing key to your application's encrypted credentials under `action_mailbox.mailgun_signing_key`, where Action Mailbox will automatically find it:

```
action_mailbox:
  mailgun_signing_key: ...
```

Alternatively, provide your Signing key in the `MAILGUN_INGRESS_SIGNING_KEY` environment variable.

Tell Action Mailbox to accept emails from Mailgun:

```
# config/environments/production.rb
config.action_mailbox.ingress = :mailgun
```

Configure Mailgun to forward inbound emails to `/rails/action_mailbox/mailgun/inbound_emails/mime`. If your application lived at `https://example.com`, you would specify the fully-qualified URL `https://example.com/rails/action_mailbox/mailgun/inbound_emails/mime`.

Mandrill

Give Action Mailbox your Mandrill API key, so it can authenticate requests to the Mandrill ingress.

Use `bin/rails credentials:edit` to add your API key to your application's encrypted credentials under `action_mailbox.mandrill_api_key`, where Action Mailbox will automatically find it:

```
action_mailbox:
  mandrill_api_key: ...
```

Alternatively, provide your API key in the `MANDRILL_INGRESS_API_KEY` environment variable.

Tell Action Mailbox to accept emails from Mandrill:

```
# config/environments/production.rb
config.action_mailbox.ingress = :mandrill
```

Configure Mandrill to route inbound emails to `/rails/action_mailbox/mandrill/inbound_emails`. If your application lived at `https://example.com`, you would specify the fully-qualified URL `https://example.com/rails/action_mailbox/mandrill/inbound_emails`.

Postfix

Tell Action Mailbox to accept emails from an SMTP relay:

```
# config/environments/production.rb
config.action_mailbox.ingress = :relay
```

Generate a strong password that Action Mailbox can use to authenticate requests to the relay ingress.

Use `bin/rails credentials:edit` to add the password to your application's encrypted credentials under `action_mailbox.ingress_password`, where Action Mailbox will automatically find it:

```
action_mailbox:
  ingress_password: ...
```

Alternatively, provide the password in the `RAILS_INBOUND_EMAIL_PASSWORD` environment variable.

Configure Postfix to pipe inbound emails to `bin/rails action_mailbox:ingress:postfix`, providing the `URL` of the Postfix ingress and the `INGRESS_PASSWORD` you previously generated. If your application lived at `https://example.com`, the full command would look like this:

```
$ bin/rails action_mailbox:ingress:postfix URL=https://example.com/rails/action_mailbox/rela
```

Postmark

Tell Action Mailbox to accept emails from Postmark:

```
# config/environments/production.rb
config.action_mailbox.ingress = :postmark
```

Generate a strong password that Action Mailbox can use to authenticate requests to the Postmark ingress.

Use `bin/rails credentials:edit` to add the password to your application's encrypted credentials under `action_mailbox.ingress_password`, where Action Mailbox will automatically find it:

```
action_mailbox:
  ingress_password: ...
```

Alternatively, provide the password in the `RAILS_INBOUND_EMAIL_PASSWORD` environment variable.

Configure Postmark inbound webhook to forward inbound emails to `/rails/action_mailbox/postmark/inbound_emails` with the username `actionmailbox` and the password you previously generated. If your application lived at `https://example.com`, you would configure Postmark with the following fully-qualified URL:

```
https://actionmailbox:PASSWORD@example.com/rails/action_mailbox/postmark/inbound_emails
```

NOTE: When configuring your Postmark inbound webhook, be sure to check the box labeled **"Include raw email content in JSON payload"**. Action Mailbox needs the raw email content to work.

Qmail

Tell Action Mailbox to accept emails from an SMTP relay:

```
# config/environments/production.rb
config.action_mailbox.ingress = :relay
```

Generate a strong password that Action Mailbox can use to authenticate requests to the relay ingress.

Use `bin/rails credentials:edit` to add the password to your application's encrypted credentials under `action_mailbox.ingress_password`, where Action Mailbox will automatically find it:

```
action_mailbox:
  ingress_password: ...
```

Alternatively, provide the password in the `RAILS_INBOUND_EMAIL_PASSWORD` environment variable.

Configure Qmail to pipe inbound emails to `bin/rails action_mailbox:ingress:qmail`, providing the `URL` of the relay ingress and the `INGRESS_PASSWORD` you previously generated. If your application lived at `https://example.com`, the full command would look like this:

```
$ bin/rails action_mailbox:ingress:qmail URL=https://example.com/rails/action_mailbox/relay/
```

SendGrid

Tell Action Mailbox to accept emails from SendGrid:

```
# config/environments/production.rb
config.action_mailbox.ingress = :sendgrid
```

Generate a strong password that Action Mailbox can use to authenticate requests to the SendGrid ingress.

Use `bin/rails credentials:edit` to add the password to your application's encrypted credentials under `action_mailbox.ingress_password`, where Action Mailbox will automatically find it:

```
action_mailbox:
  ingress_password: ...
```

Alternatively, provide the password in the `RAILS_INBOUND_EMAIL_PASSWORD` environment variable.

Configure SendGrid Inbound Parse to forward inbound emails to `/rails/action_mailbox/sendgrid/inbound_e` with the username `actionmailbox` and the password you previously generated. If your application lived at `https://example.com`, you would configure SendGrid with the following URL:

```
https://actionmailbox:PASSWORD@example.com/rails/action_mailbox/sendgrid/inbound_emails
```

NOTE: When configuring your SendGrid Inbound Parse webhook, be sure to check the box labeled **"Post the raw, full MIME message."** Action Mailbox needs the raw MIME message to work.

Examples

Configure basic routing:

```
# app/mailboxes/application_mailbox.rb
class ApplicationMailbox < ActionMailbox::Base
  routing /^save@/i     => :forwards
  routing /@replies\./i => :replies
end
```

Then set up a mailbox:

```
# Generate new mailbox
$ bin/rails generate mailbox forwards
```

```
# app/mailboxes/forwards_mailbox.rb
class ForwardsMailbox < ApplicationMailbox
  # Callbacks specify prerequisites to processing
  before_processing :require_projects

  def process
    # Record the forward on the one project, or...
    if forwarder.projects.one?
      record_forward
    else
      # ...involve a second Action Mailer to ask which project to forward into.
      request_forwarding_project
    end
  end

  private
    def require_projects
      if forwarder.projects.none?
        # Use Action Mailers to bounce incoming emails back to sender - this halts processi
        bounce_with Forwards::BounceMailer.no_projects(inbound_email, forwarder: forwarder)
      end
    end

    def record_forward
      forwarder.forwards.create subject: mail.subject, content: mail.content
    end

    def request_forwarding_project
      Forwards::RoutingMailer.choose_project(inbound_email, forwarder: forwarder).deliver_n
    end

    def forwarder
      @forwarder ||= User.find_by(email_address: mail.from)
    end
end
```

Incineration of InboundEmails

By default, an InboundEmail that has been successfully processed will be incinerated after 30 days. This ensures you're not holding on to people's data willy-nilly after they may have canceled their accounts or deleted their content. The intention is that after you've processed an email, you should have extracted all the data you needed and turned it into domain models and content on your side of the application. The InboundEmail simply stays in the system for the extra time to provide debugging and forensics options.

The actual incineration is done via the `IncinerationJob` that's scheduled to run after `config.action_mailbox.incinerate_after` time. This value is by default set to `30.days`, but you can change it in your production.rb configuration. (Note that this far-future incineration scheduling relies on your job queue being able to hold jobs for that long.)

Working with Action Mailbox in development

It's helpful to be able to test incoming emails in development without actually sending and receiving real emails. To accomplish this, there's a conductor controller mounted at `/rails/conductor/action_mailbox/inbound_emails`, which gives you an index of all the InboundEmails in the system, their state of processing, and a form to create a new InboundEmail as well.

Testing mailboxes

Example:

```
class ForwardsMailboxTest < ActionMailbox::TestCase
  test "directly recording a client forward for a forwarder and forwardee corresponding to o
    assert_difference -> { people(:david).buckets.first.recordings.count } do
      receive_inbound_email_from_mail \
        to: 'save@example.com',
        from: people(:david).email_address,
        subject: "Fwd: Status update?",
        body: <<~BODY
          --- Begin forwarded message ---
          From: Frank Holland <frank@microsoft.com>

          What's the status?
        BODY
    end

    recording = people(:david).buckets.first.recordings.last
    assert_equal people(:david), recording.creator
    assert_equal "Status update?", recording.forward.subject
    assert_match "What's the status?", recording.forward.content.to_s
  end
end
```

Please refer to the ActionMailbox::TestHelper API for further test helper methods.