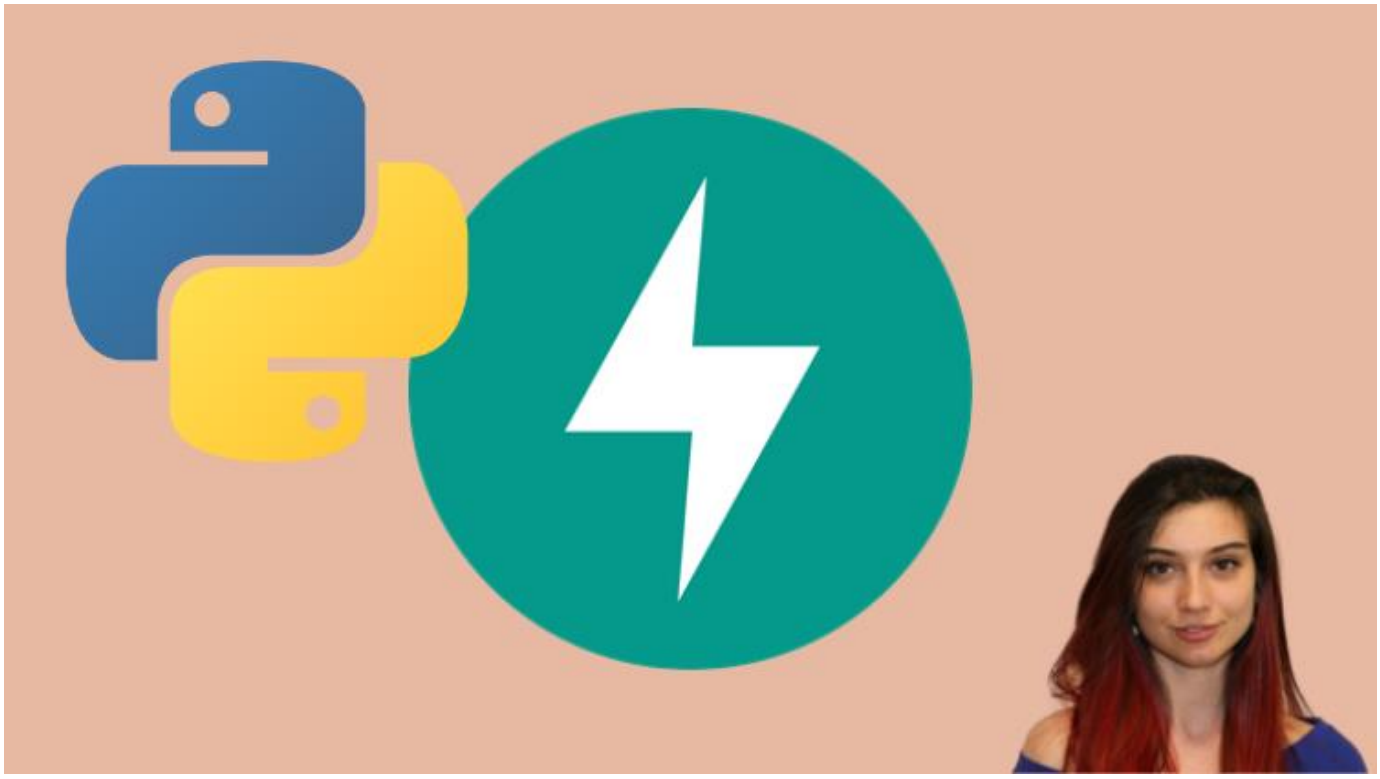
FastAPI REST – Part 7



CODE WITH FINESSE®

Ready to go?

Complaint system (course application – Part 3 – AWS Simple email service)

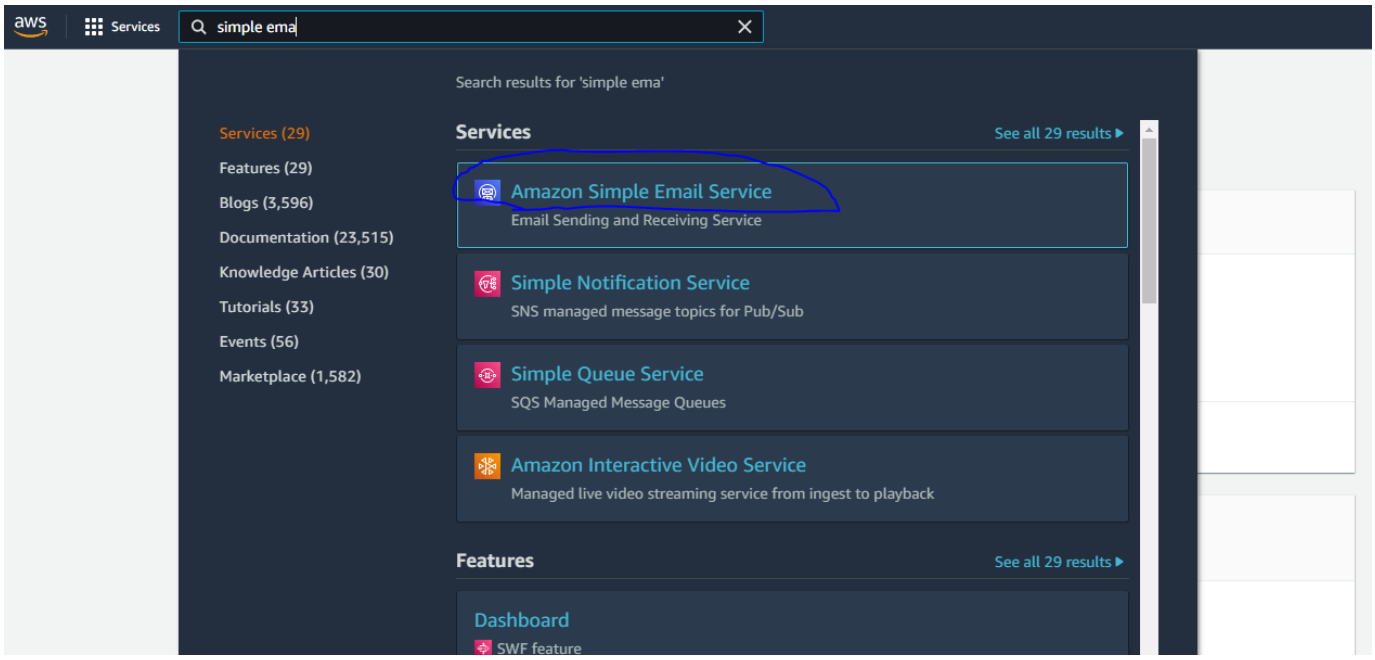
TABLE OF CONTENTS

1. Introduction

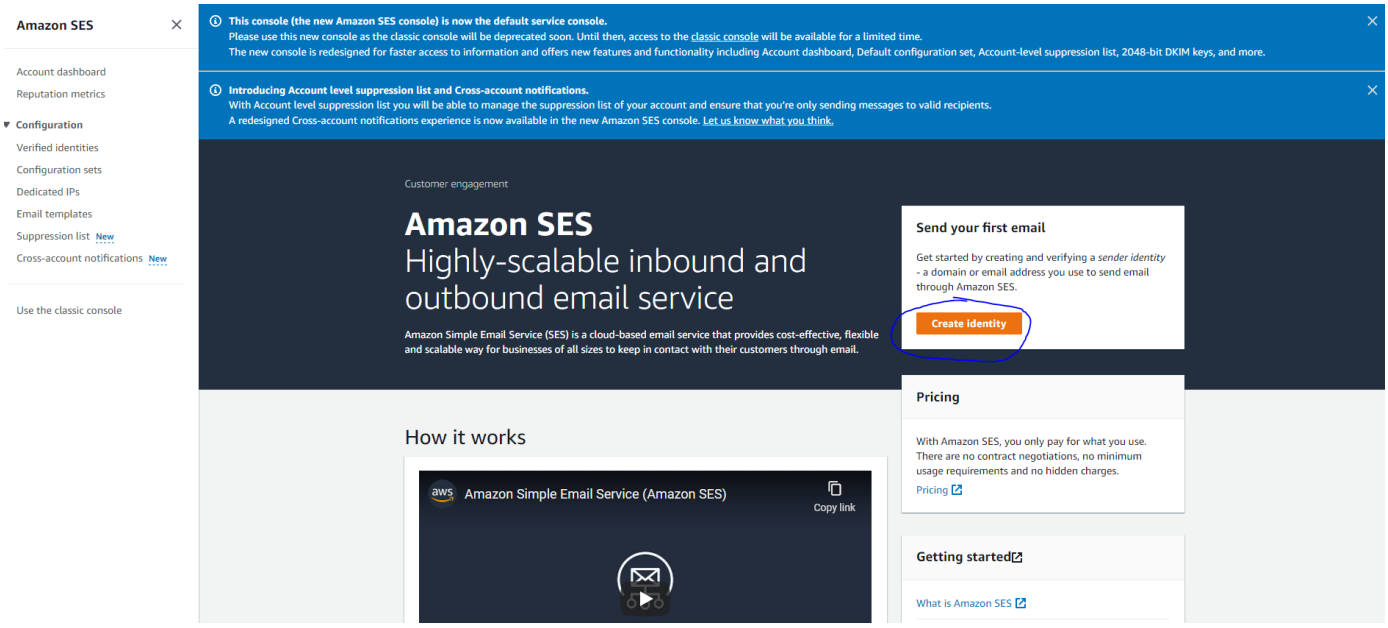
Our goal would be to integrate AWS(SES) in our application.
We will need to create an verify identity. Keep in mind we will work only with Sandbox (we will never go live, because we will have charges and we do not want that for the course application). Working only on sandbox means that every email should be listed in identities and be confirmed, so you can not send email to anyone you wish (on production you can, but again – we will use only sandbox)

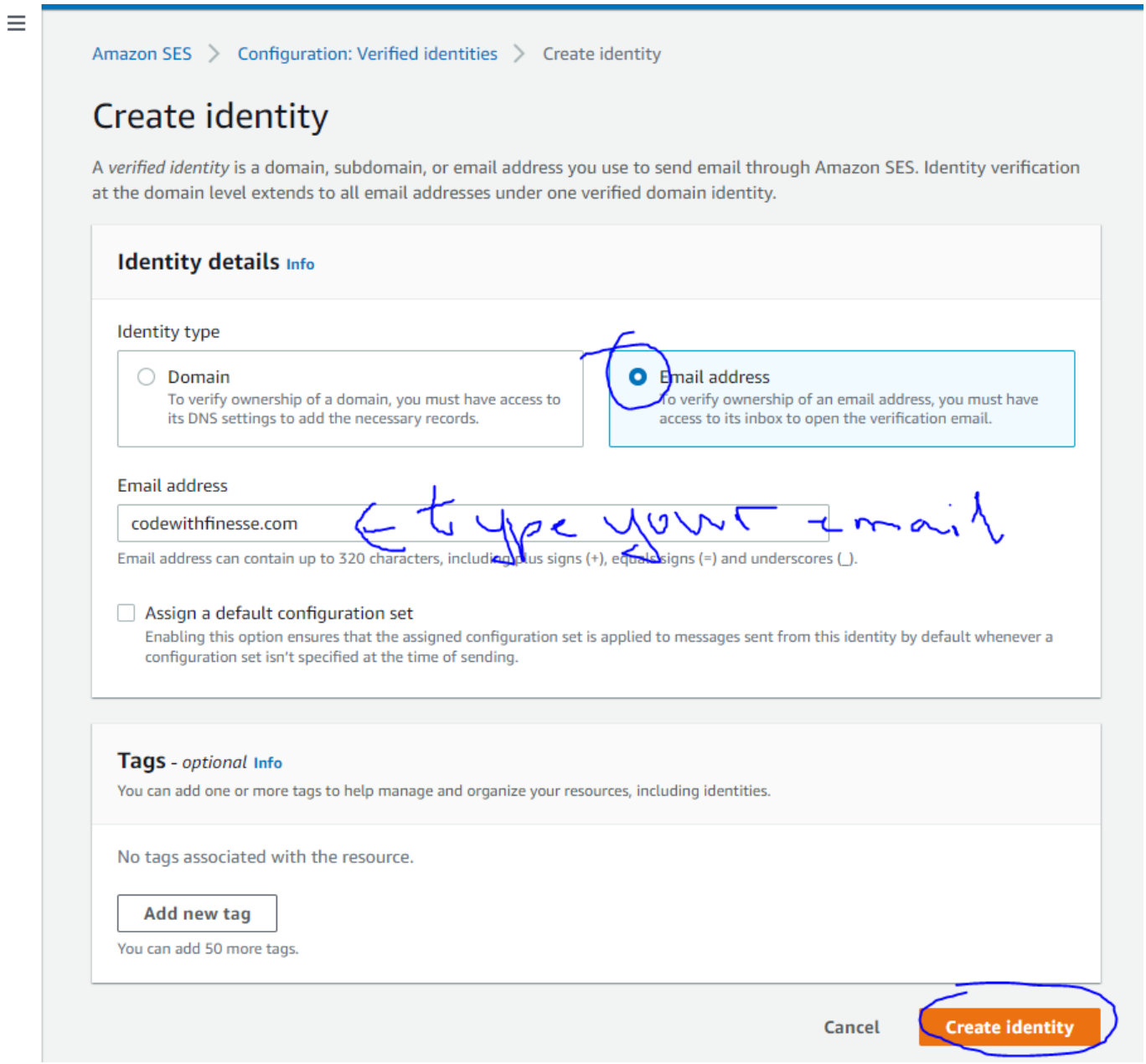
2. Set up SES

Go to the aws console and select simple email service:

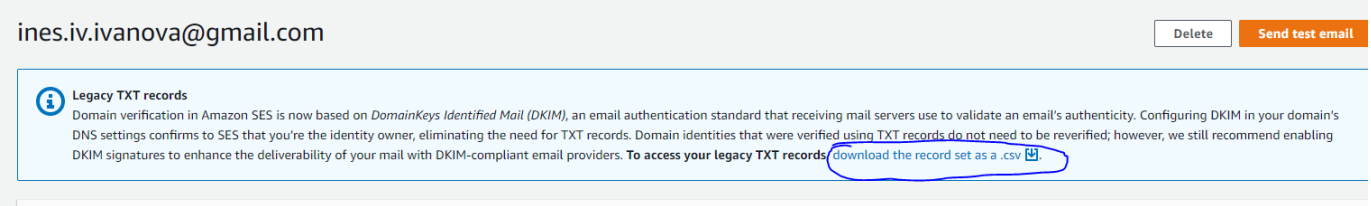


Then we need to create identity. Please choose “Email” option:

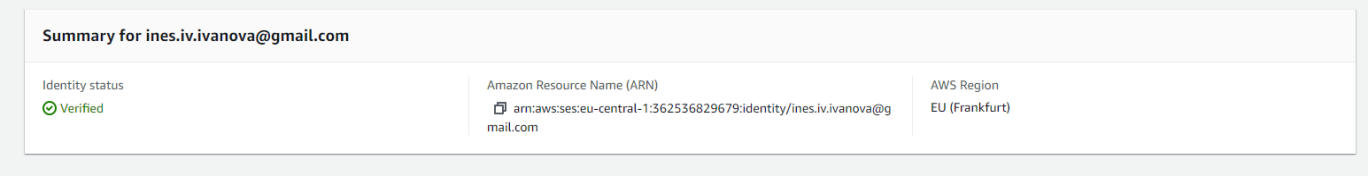




Here you should type your email and you have to have access to inbox of this email.
You will not receive any confirmation email, until you “download the record as a .csv” file option is clicked



Next go to your sandbox and click the link from the email that AWS has sent to you, so that you can confirm your identity. If everything works you should see something like this:



3. Integrate

Once you have confirmed your email, you can do the following - create a file “ses.py” user “services” and place the following code:

```
from decouple import config
import boto3


class SESService:
    def __init__(self):
        secret_id = config("AWS_ACCESS_KEY")
        secret_key = config("AWS_SECRET")

        self.ses = boto3.client(
            "ses",
            region_name=config("SES_REGION"),
            aws_access_key_id=secret_id,
            aws_secret_access_key=secret_key,
        )

    def send_mail(self, subject, to_addresses, text_data):
        body = {}
        body.update({"Text": {"Data": text_data, "Charset": "UTF-8"}})

        try:
            self.ses.send_email(
                Source= "codewithfinesse@gmail.com",
                Destination={
                    "ToAddresses": to_addresses,
                    "CcAddresses": [],
                    "BccAddresses": [],
                },
                Message={
                    "Subject": {"Data": subject, "Charset": "UTF-8"},
                    "Body": body,
                },
            )
        except Exception as ex:
            raise ex
```

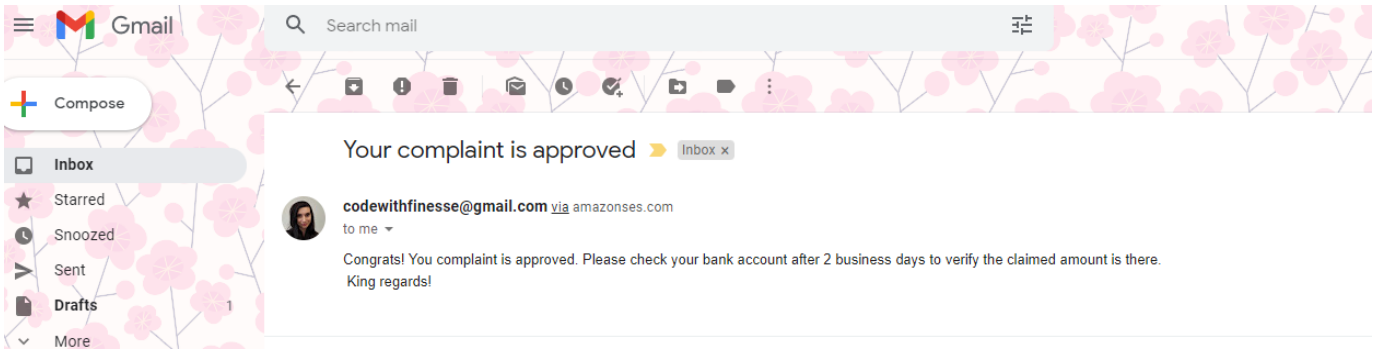
You should define a variable SES_REGION in your .env file. You can see the region from up right corner of aws account. The access_key and the secret are the same from the previous part – you reuse the same credentials for both services.

Now we can add the following logic to the ComplaintManager class:

```
@staticmethod
async def approve(id_):
    await database.execute(complaint.update().where(complaint.c.id ==
id_).values(status=State.approved))
    ses.send_mail("Your complaint is approved", ["ines.iv.ivanova@gmail.com"],
"Congrats! You complaint is approved. Please check your bank account after 2
business days to verify the claimed amount is there.\n King regards!")
```

You have to change the email ines.iv.ivanova@gmail.com with your verified email in AWS. Here of course we would like to send the email to the complainer, not every time to the same email, but you have to make sure you create a user with this email, make a complaint as this user, and then make request as approver to approve this created claim. Then you can replace the email with
user. Email
where user would be passed to the method from the resource function and fetched from request state.

If you test it you should see something like this in your inbox:



Now you can try to add similar logic to the reject's function, with different text.