How to handle lots of requests and not get crazy along the way

- Damian Kampik, u2i

Short intro

- 5 yrs of experience as Ruby programmer
- 1,5 yrs at u2i
- We’re just around the corner - come visit us!

u2i

QR Codes

AGENDA

Agenda

1. What's the problem?
2. How did we solve it?
    a. The idea
    b. Lambda
    c. Sidekiq
    d. Apex
3. Summary

PROBLEM

What did we need to implement?

What did we need to implement?

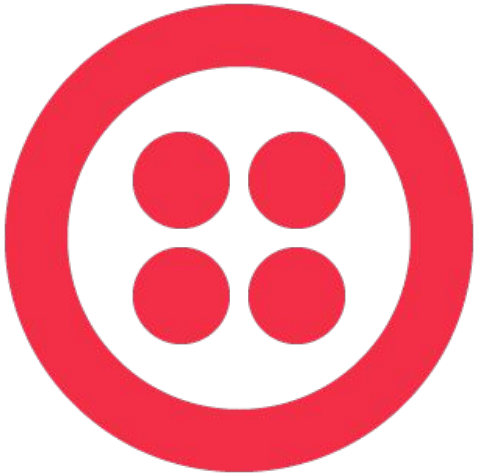
- SMS emergency alerts

What did we need to implement?

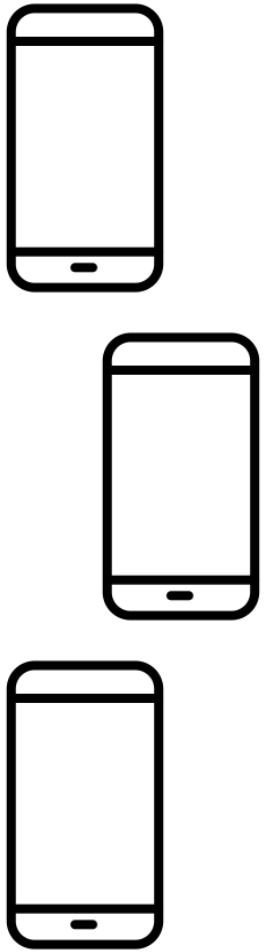
- SMS emergency alerts
  - Maximize the number of successfully received messages

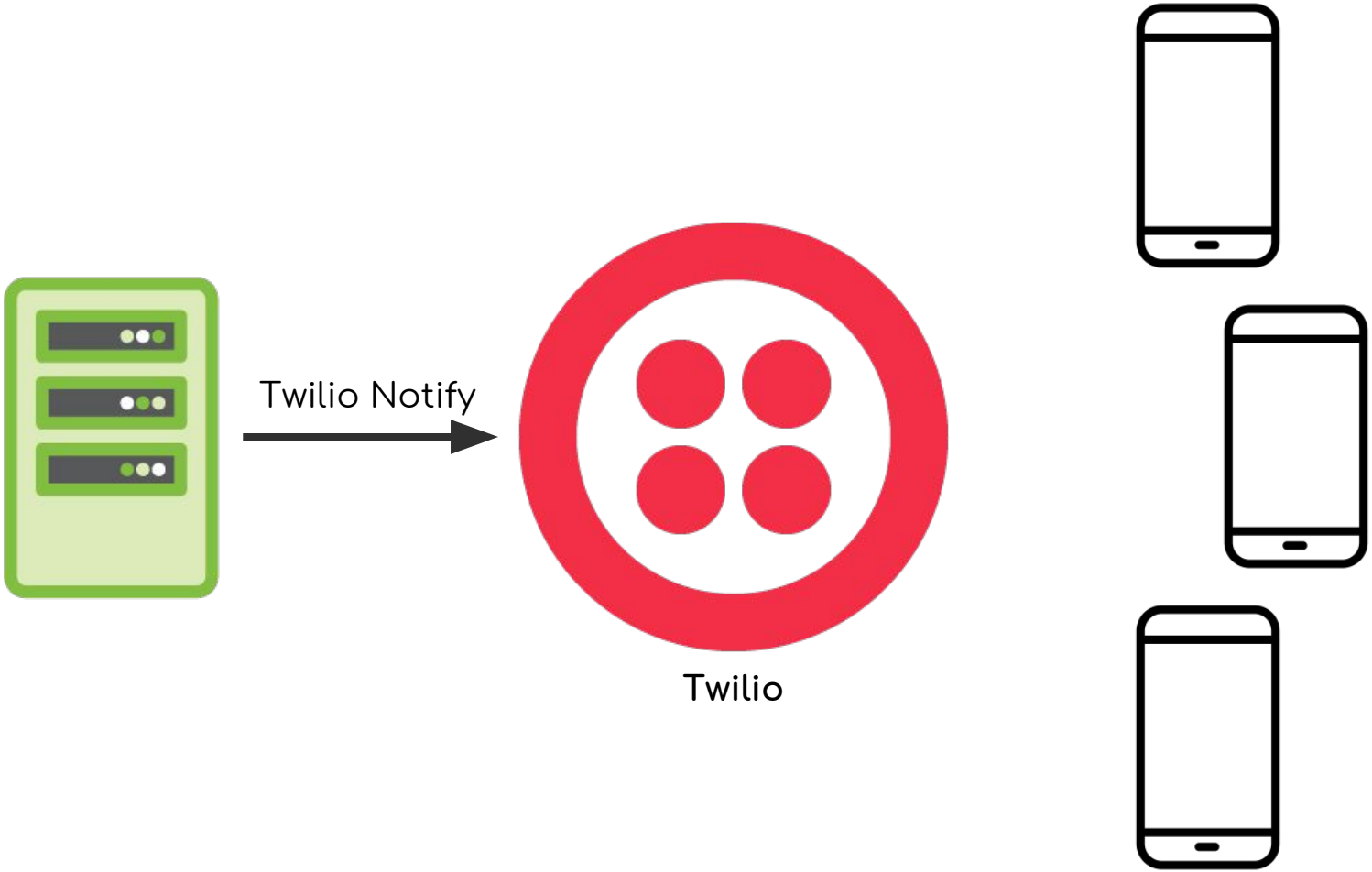
What did we need to implement?

- SMS emergency alerts
  - Maximize the number of successfully received messages
  - Collect delivery statistics



Twilio




Twilio Notify
Twilio

Twilio Notify

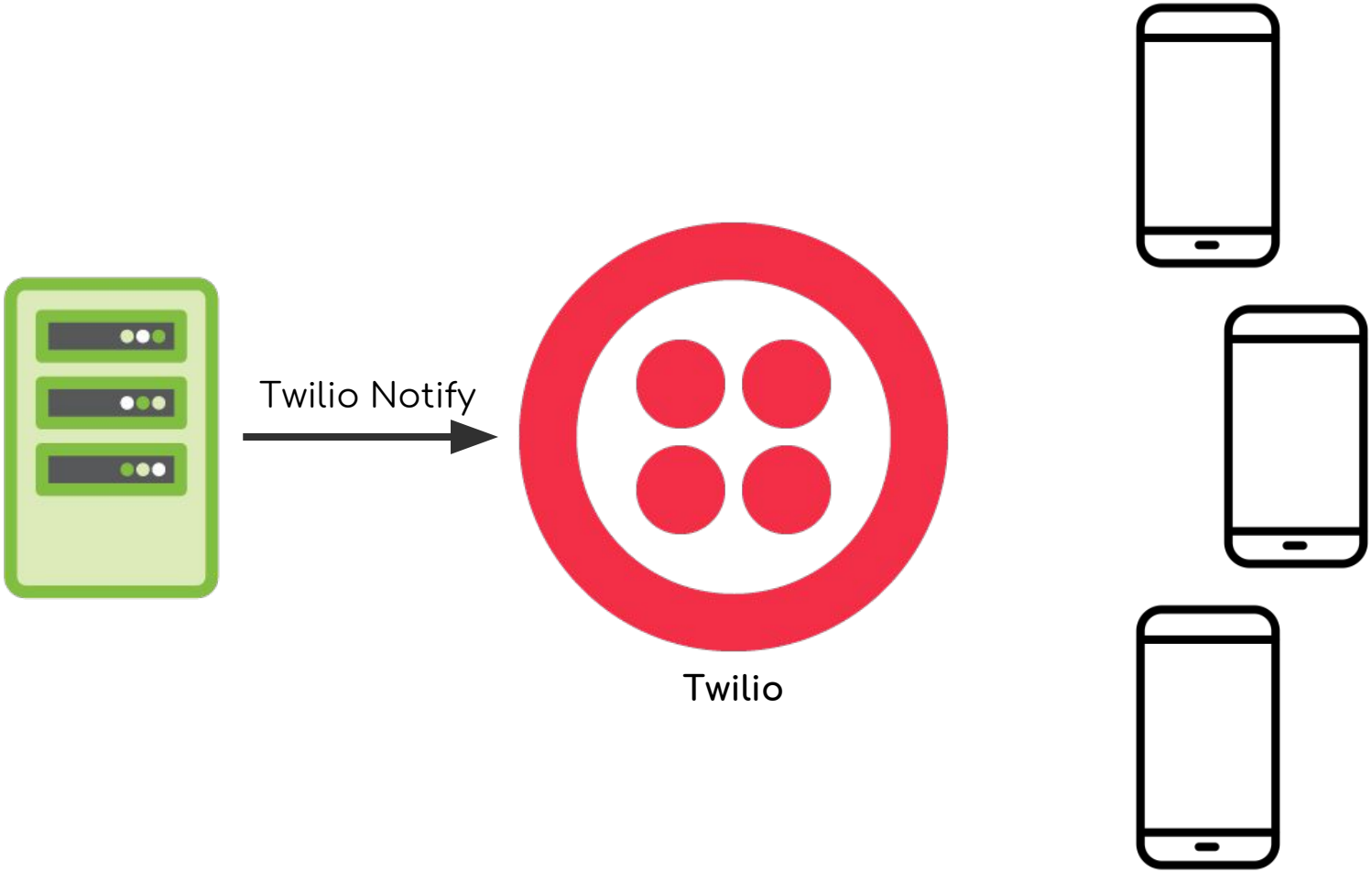
```
def service
  return @service unless @service.nil?

  account_sid = ENV['TWILIO_ACCOUNT_SID']
  auth_token = ENV['TWILIO_AUTH_TOKEN']
  client = Twilio::REST::Client.new(account_sid, auth_token)
  @service = client.notify.v1
             .services(ENV['TWILIO_NOTIFY_SERVICE'])
end
```

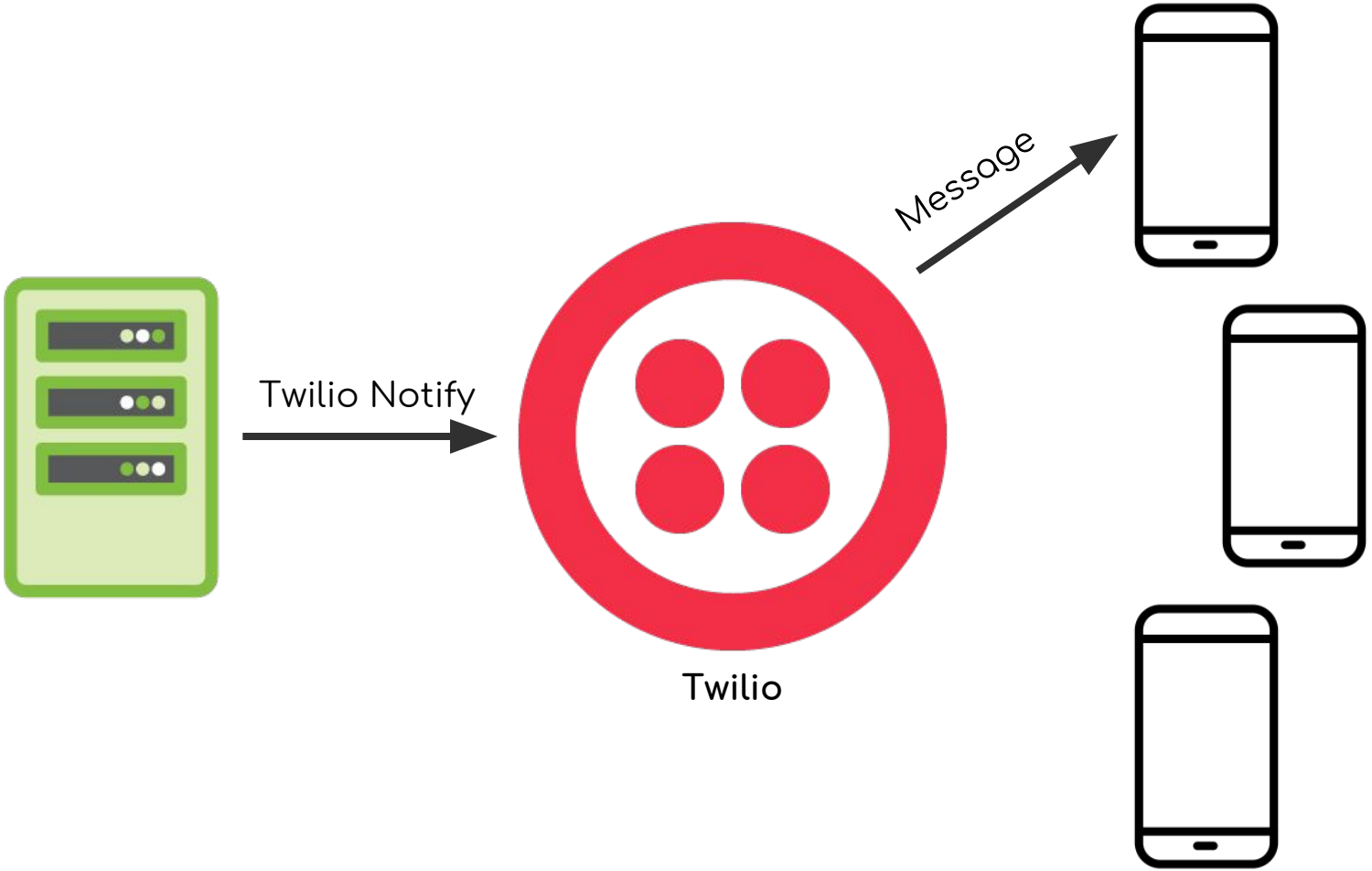
Twilio Notify

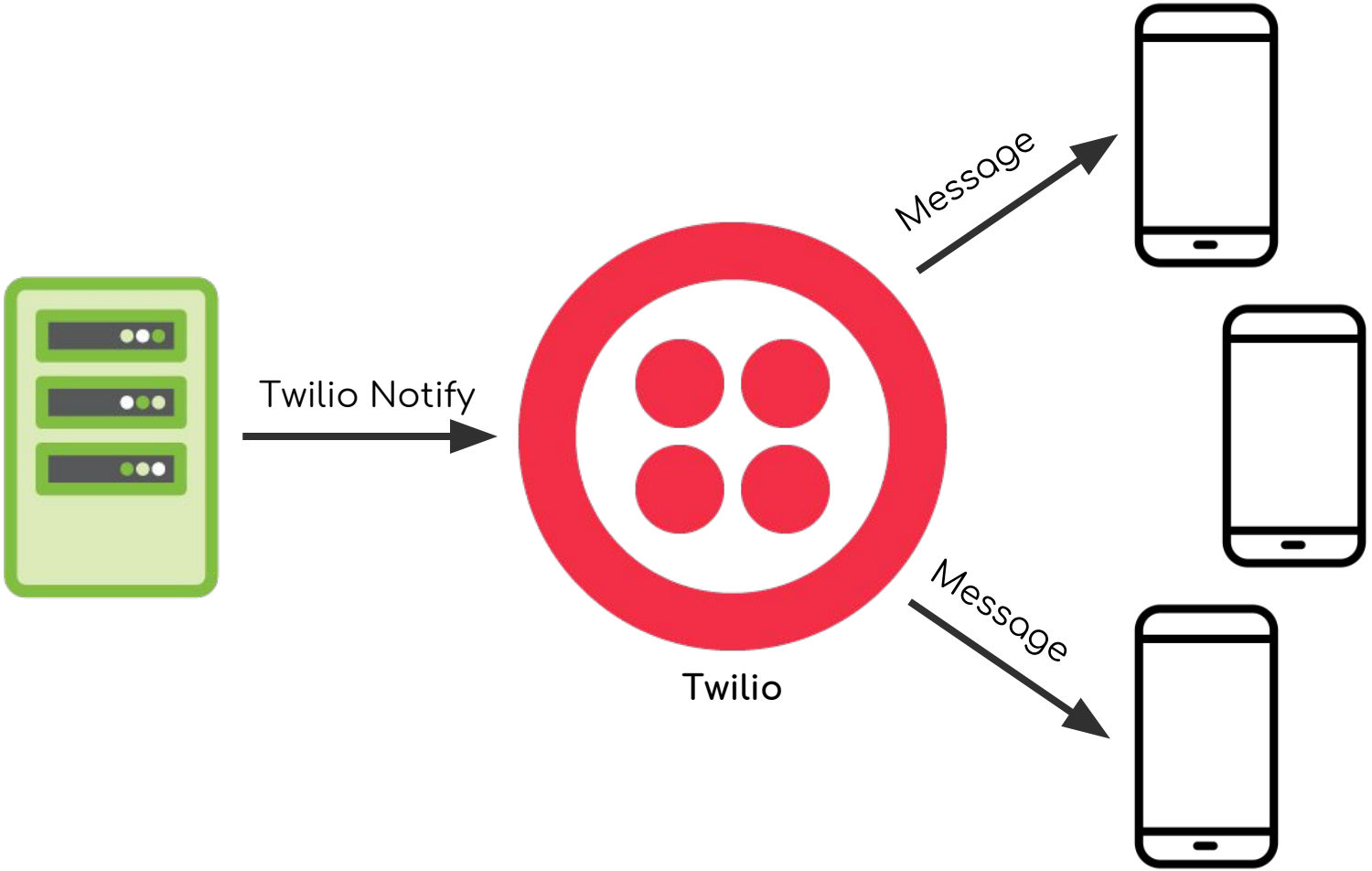
```
def phones_binding
  @phones.map do |phone|
    {
      binding_type: :sms,
      address: phone
    }.to_json
  end
end
```

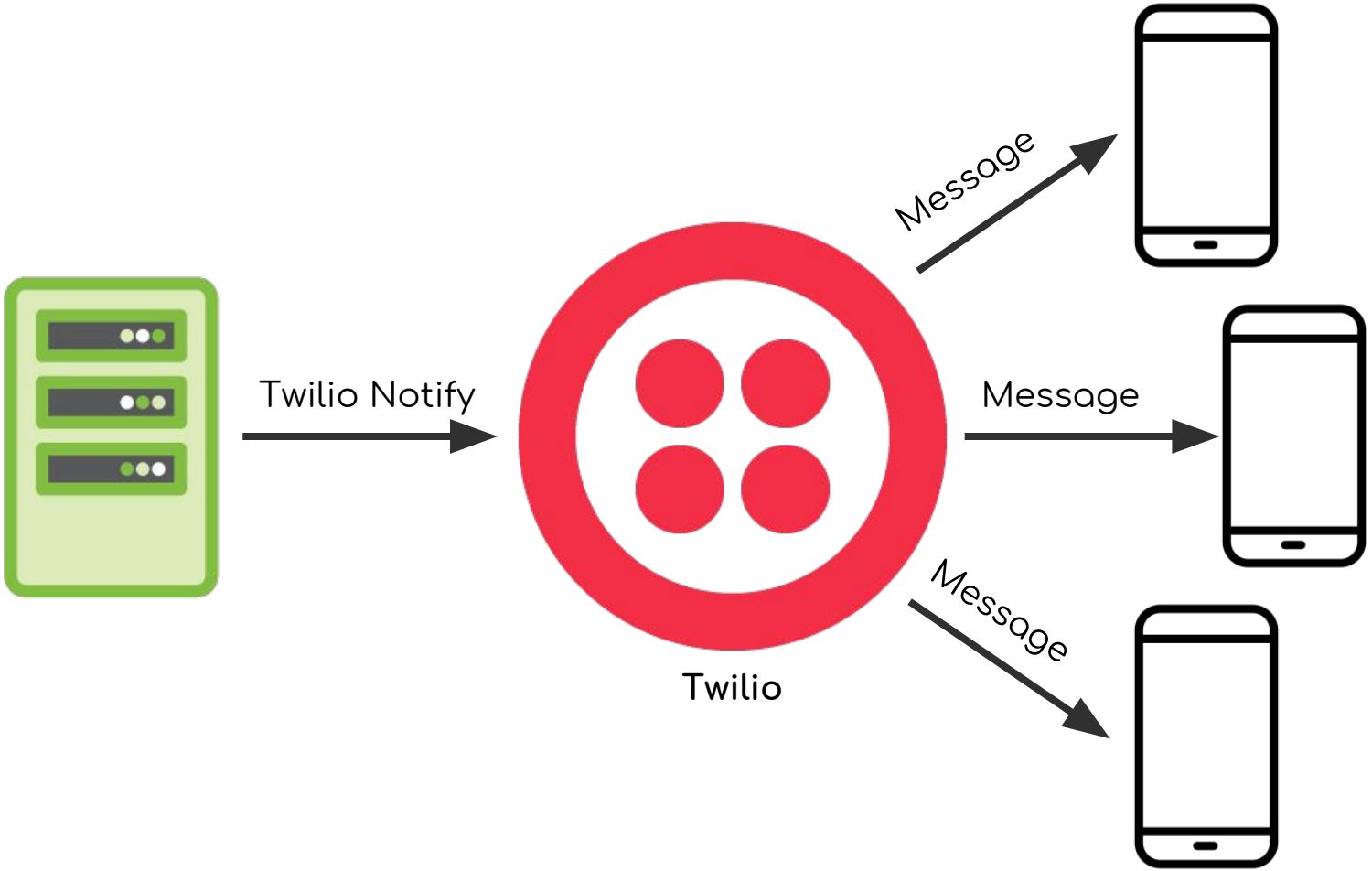
Twilio Notify

```
def send_message
  service.notifications.create({
    to_binding: phones_binding,
    body: @content
  })
end
```


Twilio Notify
Twilio


Message
Twilio Notify
Twilio


Twilio Notify
Twilio
Message
Message


Message
Twilio Notify
Message
Message
Twilio

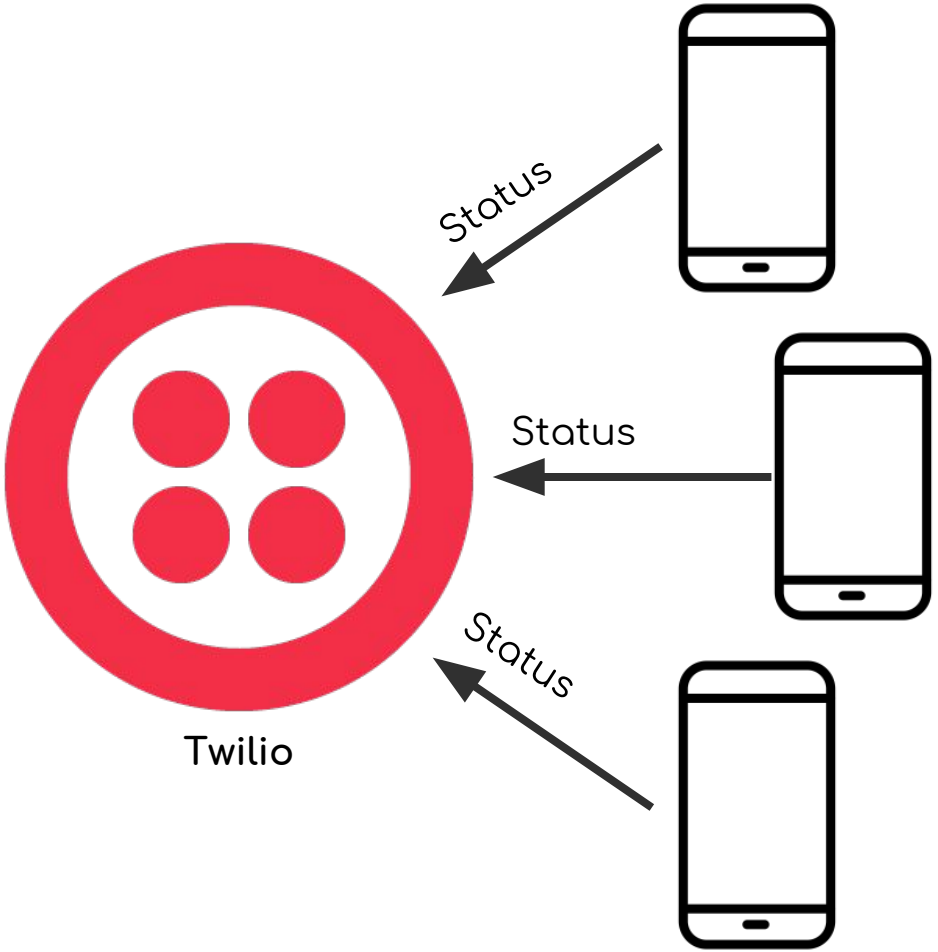
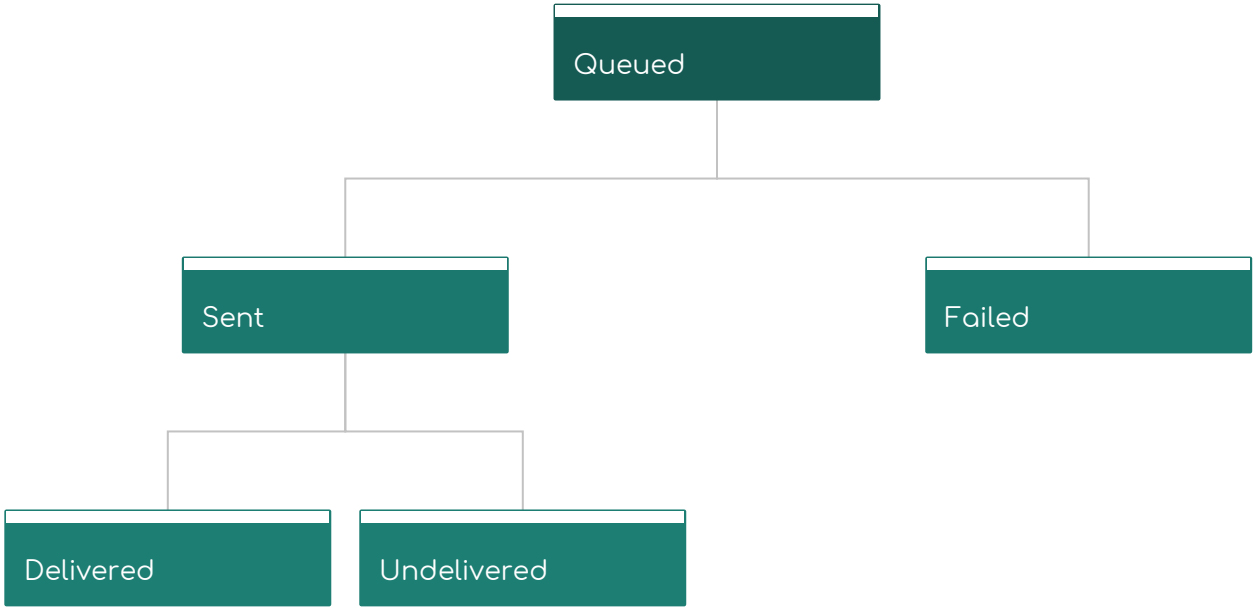
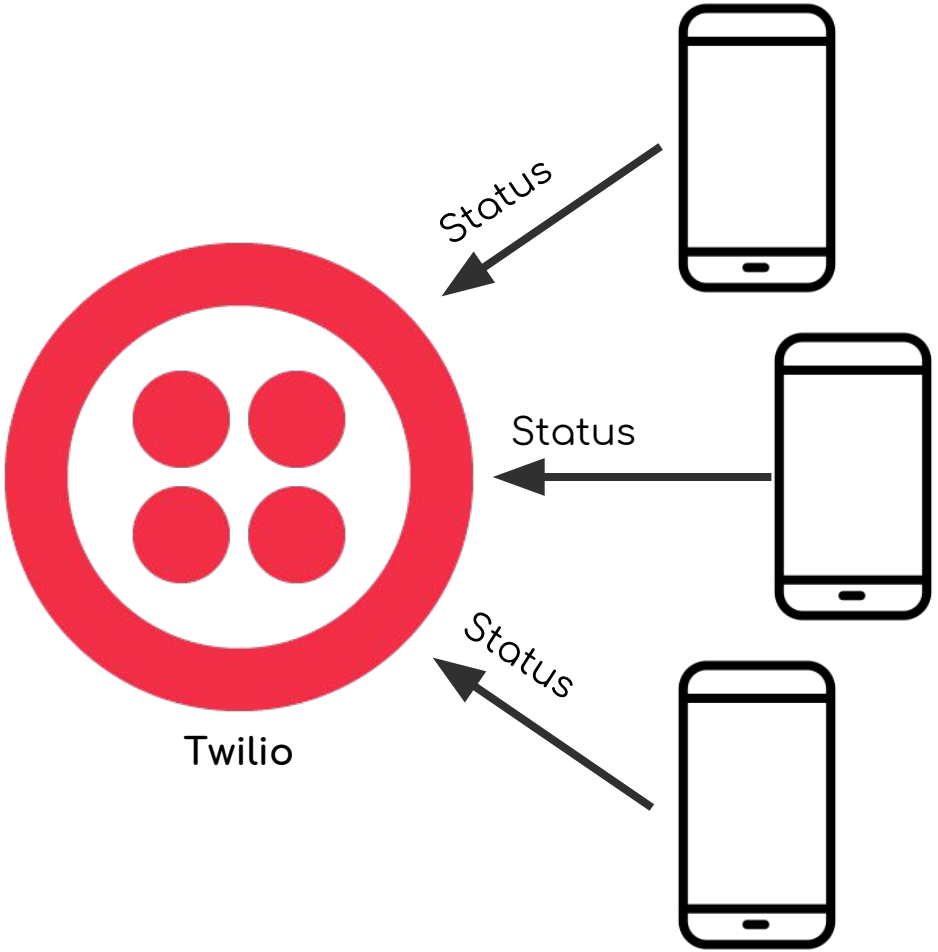
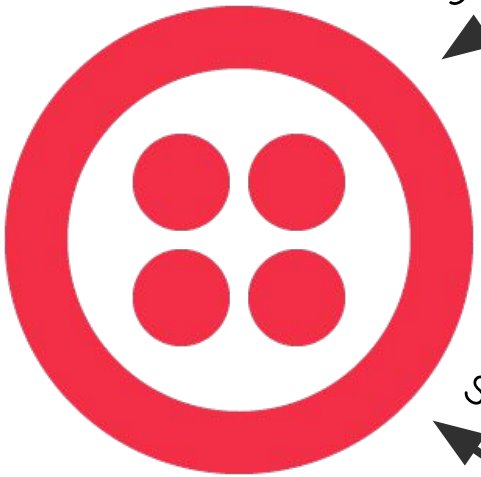

Status
Status
Status
Twilio

Diagram of message statuses



Status callbacks docs


Status
Status
Status
Twilio

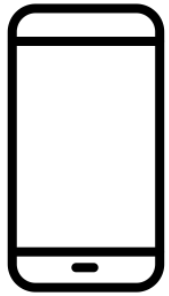


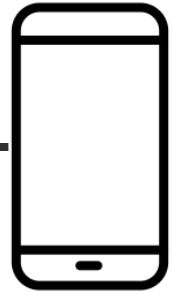
Twilio

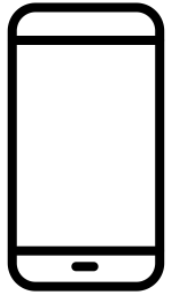
Status x3

Status x3

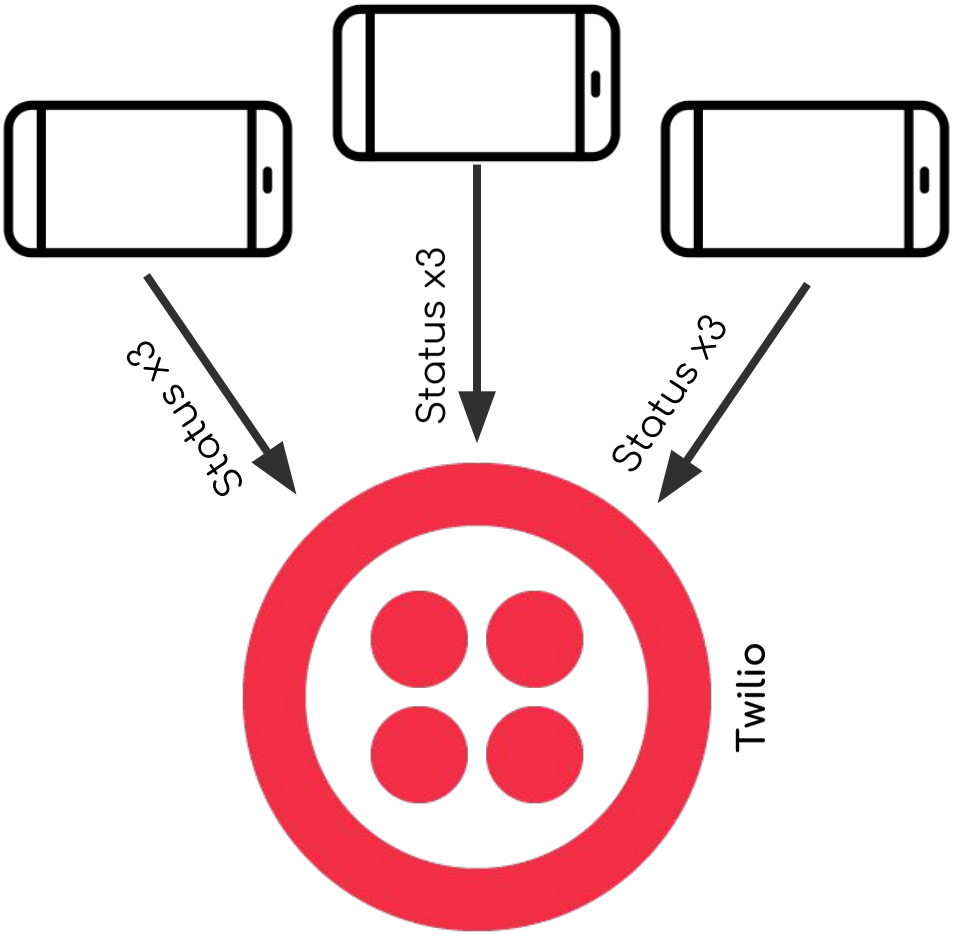
Status x3

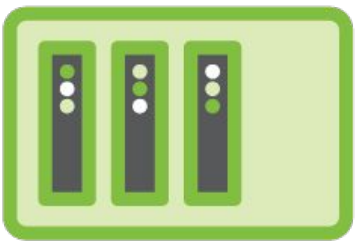




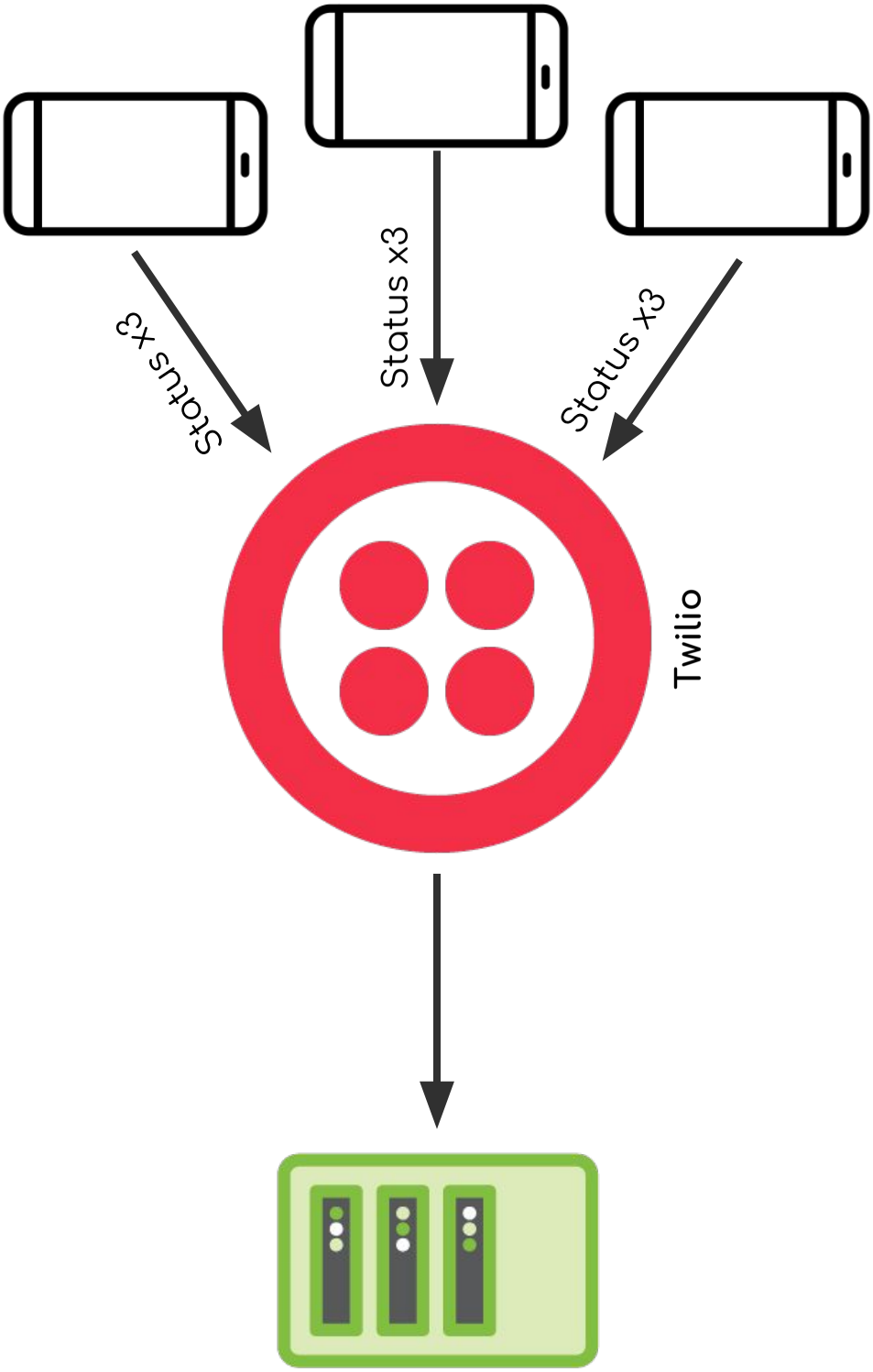


+ 2997 remaining recipients

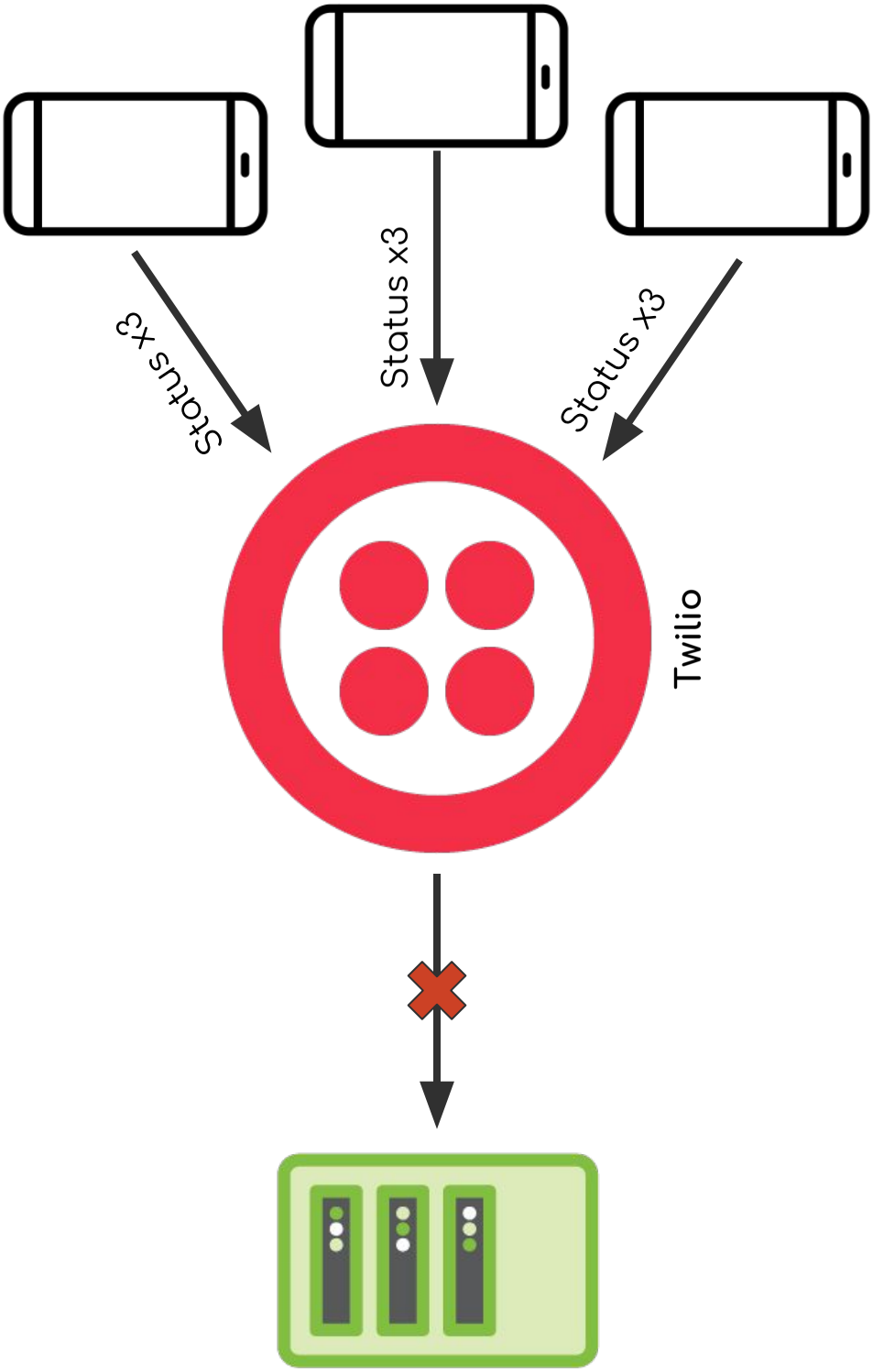




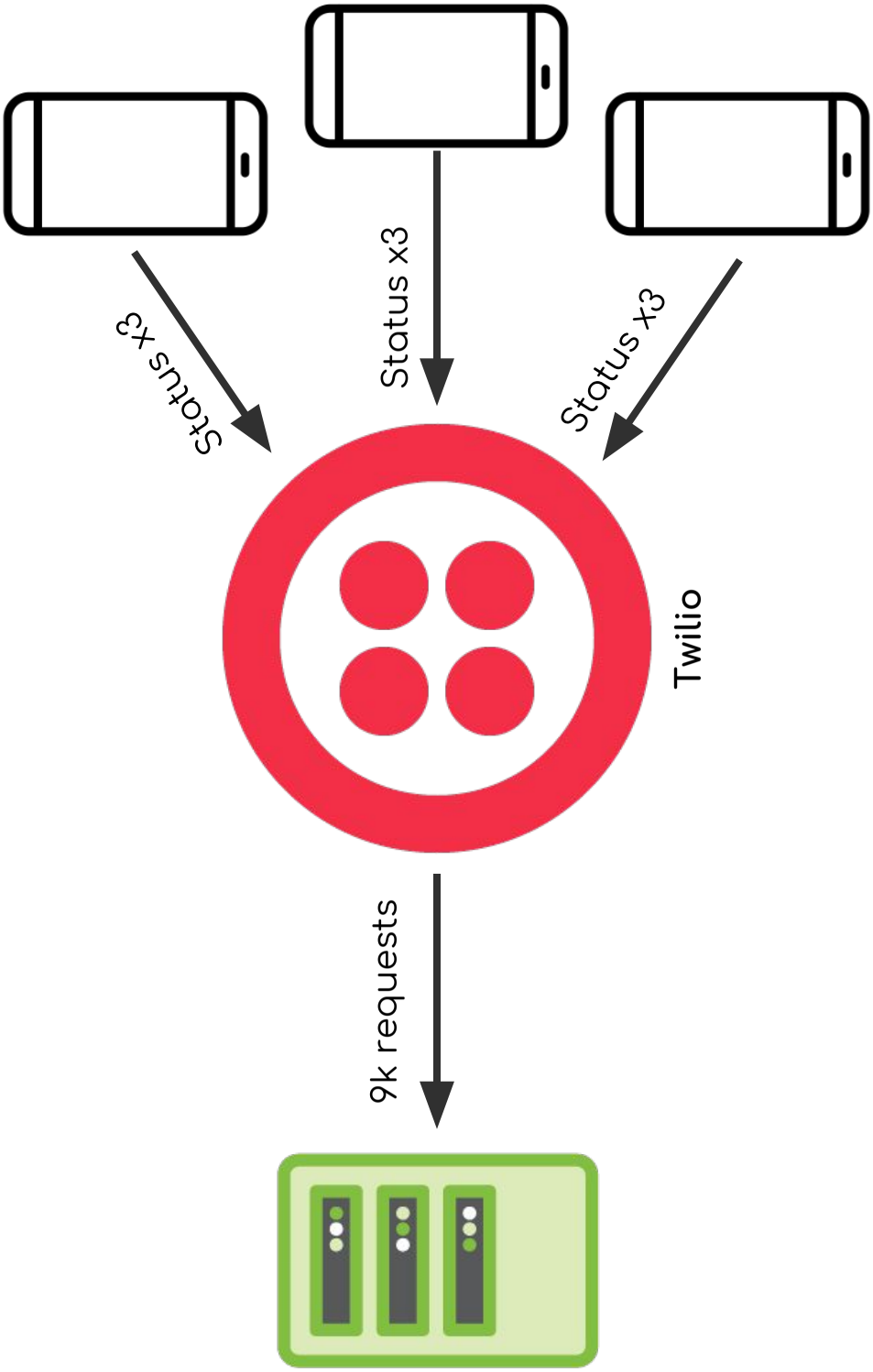
+ 2997 remaining recipients

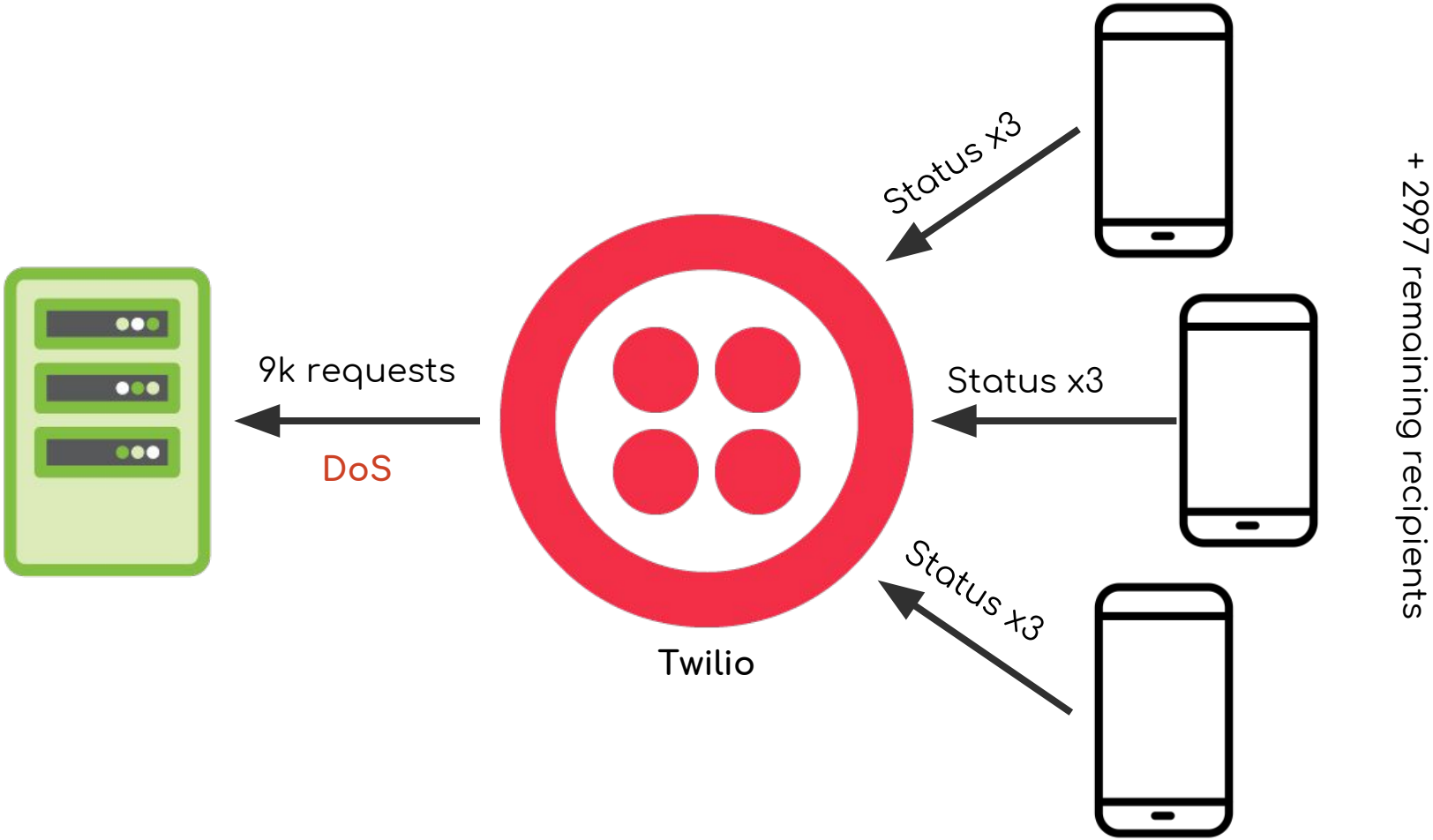


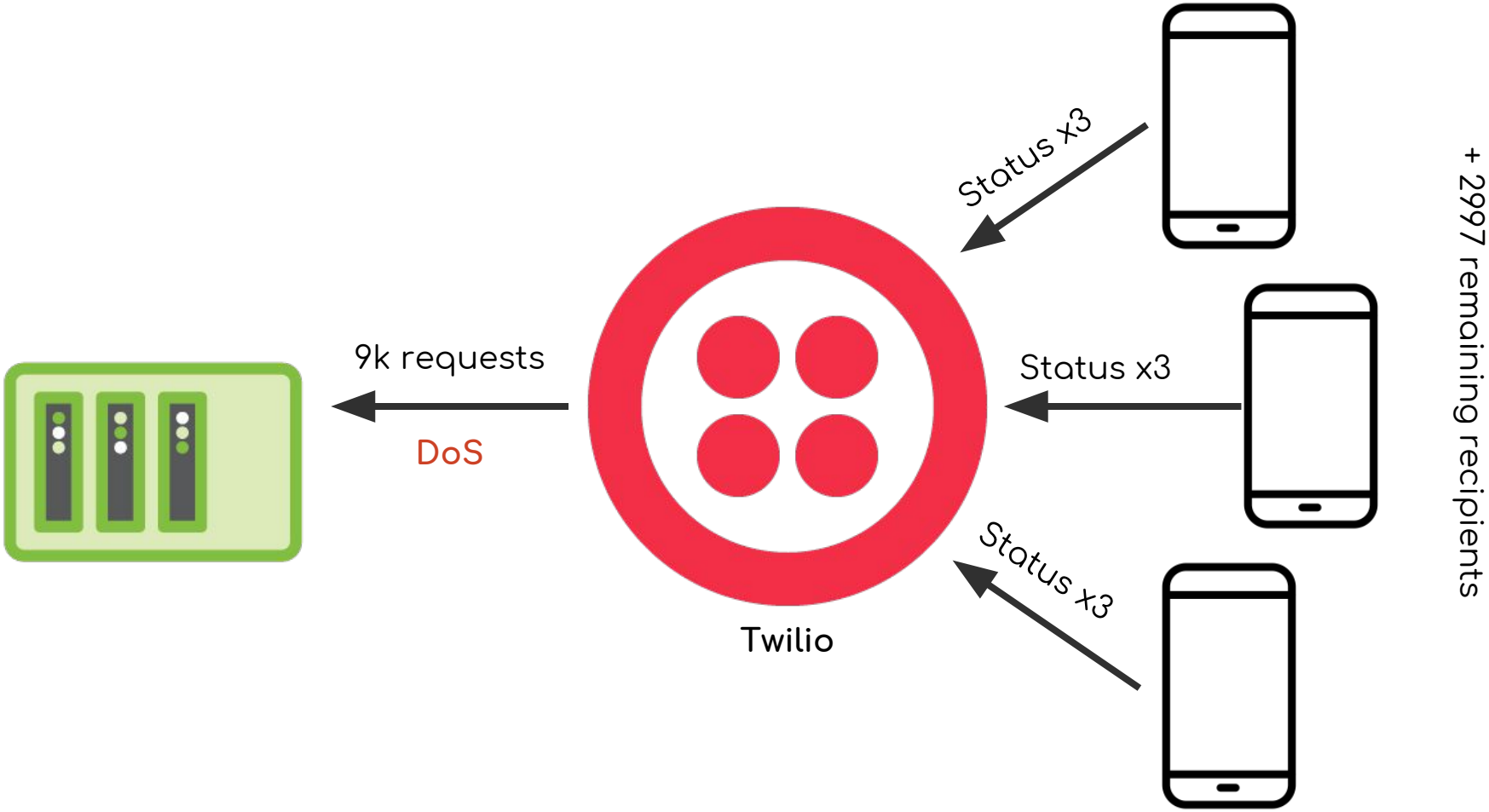
+ 2997 remaining recipients



+ 2997 remaining recipients




Status x3
Status x3
Status x3
+ 2997 remaining recipients
9k requests
DoS
Twilio


Status x3
Status x3
Status x3
+ 2997 remaining recipients
9k requests
DoS
Twilio

9k requests

DoS

Twilio

Status x3

Status x3

Status x3

+ 2997 remaining recipients

Thx, Twilio :(

What was the problem?

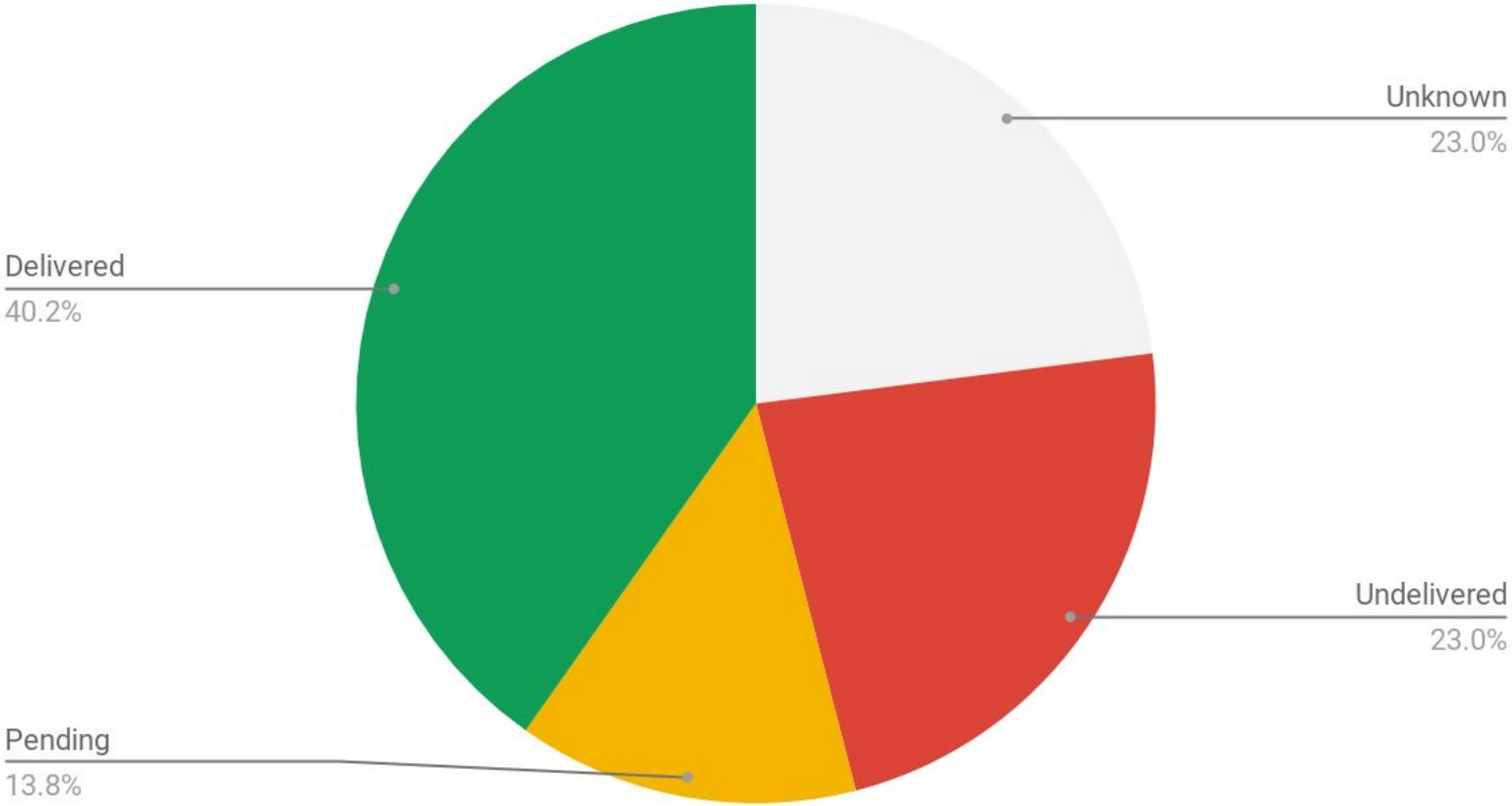
What was the problem?

- Too many requests from Twilio interfering with regular website traffic

What was the problem?

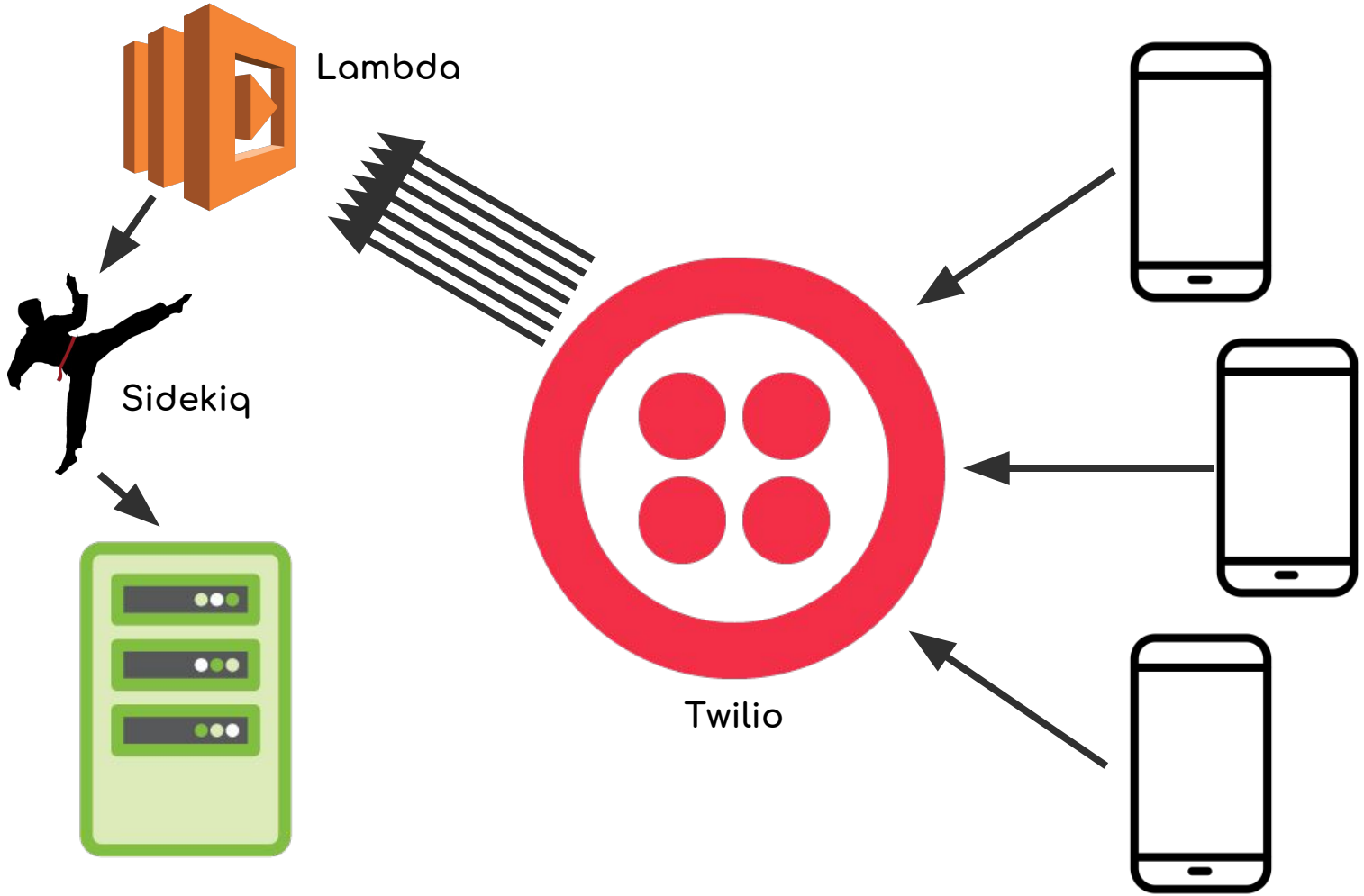
- Too many requests from Twilio interfering with regular website traffic
- We could not display correct delivery statistics

Message delivery rate



SOLUTION

AWS Lambda


Lambda
Sidekiq
Twilio

Why Lambda?

Why Lambda?

- Load on AWS side

Why Lambda?

- Load on AWS side
- High scalability - up to 1000 simultaneous executions

Why Lambda?

- Load on AWS side
- High scalability - up to 1000 simultaneous executions
- We’re paying for the infrastructure only when we have to handle incoming Twilio requests

Why Lambda?

- Load on AWS side
- High scalability - up to 1000 simultaneous executions
- We’re paying for the infrastructure only when we have to handle incoming Twilio requests
- The feature could be written in a few simple methods

Why Sidekiq?

Why Sidekiq?

- To enqueue status update data, you only have to create a record with job metadata in Redis

Why Sidekiq?

- To enqueue status update data, you only have to create a record with job metadata in Redis
- Servers will be handling status updates when it has free resources

Why Sidekiq?

- To enqueue status update data, you only have to create a record with job metadata in Redis
- Servers will be handling status updates when it has free resources
- You can assign workers to process message status updates exclusively

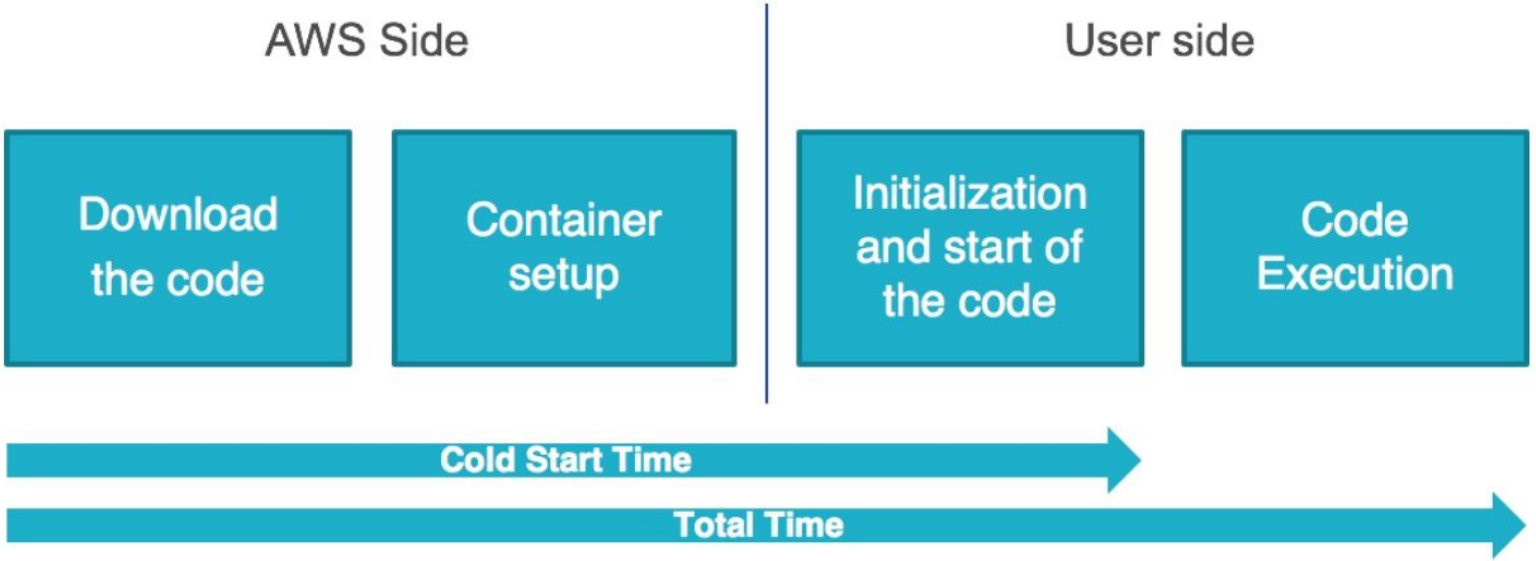
Why Sidekiq?

- To enqueue status update data, you only have to create a record with job metadata in Redis
- Servers will be handling status updates when it has free resources
- You can assign workers to process message status updates exclusively

What have we learnt the hard way?

What have we learnt the hard way?

Cold start - behind the scenes



Source : AWS re:Invent 2017 – Become a serverless Black Belt

How to bypass the issue?

How to bypass the issue?

- Declare lowest amount of RAM possible

How to bypass the issue?

- Declare lowest amount of RAM possible
- Use interpreted language

How to bypass the issue?

- Declare lowest amount of RAM possible
- Use interpreted language
- Optimize memory consumption of executed code

How to bypass the issue?

- Declare lowest amount of RAM possible
- Use interpreted language
- Optimize memory consumption of executed code
- Optimize size of the code

IMPLEMENTATION

Twilio request validation

```
def handle(event, context):
  if 'headers' in event and 'X-Twilio-Signature' in
    event['headers'] and 'body' in event:
    twilio_signature =
      event['headers']['X-Twilio-Signature']
    form_parameters = dict(parse_qsl(event['body']))

    if request_valid(form_parameters, twilio_signature)
      and u'From' in form_parameters:
...
```

Twilio request validation

```
def request_valid(parameters, signature):
  validator =
    RequestValidator(os.environ['TWILIO_AUTH_TOKEN'])

  return validator.validate(
    os.environ['REQUEST_URL'],
    parameters,
    signature
  )
```

Sidekiq thru Python

```
def push_to_sidekiq(parameters):
  elasticache = redis.StrictRedis(
    host=os.environ['ELASTICACHE_URL'],port=6379, db=0)

  args = [{
    "job_class": "StatusCallbackJob",
    "job_id": str(uuid.uuid4()),
    "queue_name": "callbacks",
    "locale": "en",
    "arguments": [parameters['To'], parameters['MessageStatus'],
                  parameters['Body'], int(time.time())]
  }]
```

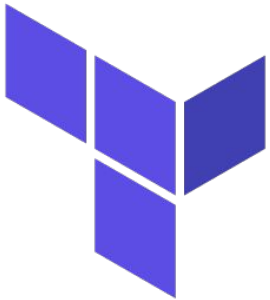
Sidekiq thru Python

```
sidekiq_params = {
  "class": "ActiveJob::QueueAdapters::SidekiqAdapter::JobWrapper",
  "wrapped": "StatusCallbackJob",
  "jid": binascii.b2a_hex(os.urandom(12)),
  "args": args,
  "created_at": int(time.time()),
  "enqueued_at": int(time.time()),
  "queue": "callbacks",
  "retry": True
}
elasticache.lpush("{0}:queue:callbacks".format(
  os.environ['REDIS_NAMESPACE']),json.dumps(sidekiq_params))
```

Worker collecting the data

```
class StatusCallbackJob < ActiveJob::Base
  queue_as :callbacks

  VALID_ALERT_THRESHOLD = 6.hours
  TWILIO_STATUSES = %w[queued failed sent delivered
undelivered].freeze

  def perform(phone, status, content, timestamp)
    ...
  end
end
```

A quick note about Terraform



HashiCorp
Terraform

Lambda in Terraform tutorial

Apex - efficient Lambda management

Apex website

Project structure

```
.
├── env.json
├── functions
│   └── alerts
│       ├── <libraries>
│       ├── main.py
│       ├── requirements.txt
│       └── setup.cfg
├── <terraform related files>
└── project.json
```

Project structure

```
.
├── env.json
├── functions
│   └── alerts
│       ├── <libraries>
│       ├── main.py
│       ├── requirements.txt
│       └── setup.cfg
├── <terraform related files>
└── project.json
```

Project structure

```
.
├── env.json
├── functions
│   └── alerts
│       ├── <libraries>
│       ├── main.py
│       ├── requirements.txt
│       └── setup.cfg
├── <terraform related files>
└── project.json
```

Project structure

```
.
├── env.json
├── functions
│   └── alerts
│       ├── <libraries>
│       ├── main.py
│       ├── requirements.txt
│       └── setup.cfg
├── <terraform related files>
└── project.json
```

project.json

```
{
  "name": "alerts",
  "nameTemplate": "{{ .Project.Environment }}_{{
.Function.Name }}",
  "description": "",
  "memory": 128,
  "timeout": 60,
  "role": <your IAM role>
...
```

project.json

```
  "environment": {
    "REQUEST_URL": <API Gateway URL to invoke the function>,
    "ELASTICACHE_URL": <Redis instance URL>,
    "REDIS_NAMESPACE": <Redis namespace declared in Sidekiq
config>
  },
  "vpc": {
    "securityGroups": ["sg-1234567890abcdef"],
    "subnets": ["subnet-12345678", "subnet-90abcdef"]
  }
...
```

project.json

```
  "hooks": {
    "build": "pip install -r requirements.txt -t ."
  }
}
```

How to use Apex

```
$ apex deploy --env=<target_env> --env-file=<env_file>
<function_name>
```

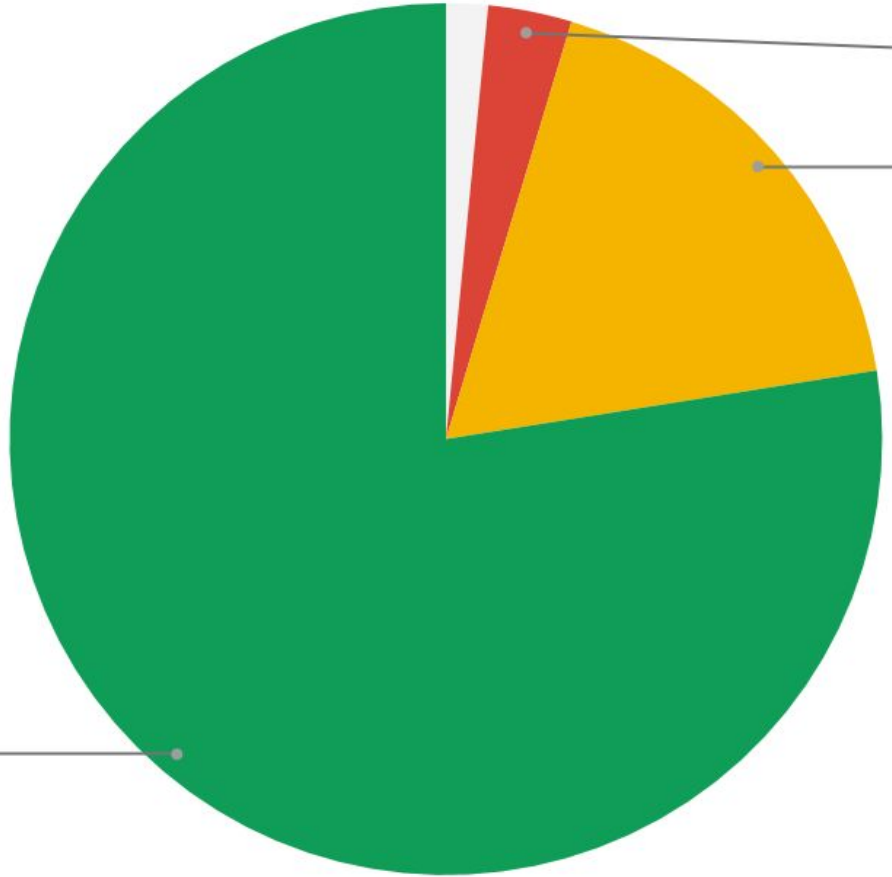
How to use Apex

```
$ apex deploy --env=<target_env> --env-file=<env_file>
<function_name>
```

```
$ apex invoke --env=<target_env> --env-file=<env_file>
<function_name> < event.json
```


LAMBDA
SIDEKIQ
TWILIO
RAILS
APEX

Message delivery rate



TAKEAWAYS

What to remember?

What to remember?

- The architecture does not scale indefinitely

What to remember?

- The architecture does not scale indefinitely
- You can fall victim to cold start

What to remember?

- The architecture does not scale indefinitely
- You can fall victim to cold start
- This solution will work only for scenarios in which you want to throttle the traffic

What to remember?

- The architecture does not scale indefinitely
- You can fall victim to cold start
- This solution will work only for scenarios in which you want to throttle the traffic
- From Nov 2018 you can use Ruby on Lambda!

QUESTIONS?

Thanks a lot! :)