M3RCURY

yet another coding challenge

UBER

presented by prisco napoli

AGENDA

- About me
- The Challenge
- Architecture
- Demo
- Future Work
- Q&A

ABOUT ME

Who in the world am I? Ah, that's the great puzzle.

I work in Corvil, a machine-time analytics company based in Dublin

Design and develop features in C++/Java for:

- Analytics Plugin (real-time decoders for Financial, Middleware and Enterprise protocols)
- Security Analytics (real-time threats detection, e.g. DNS tunnelling, watchlist, Emerging Threats, CarbonBlack ecc.)
- Connectors (connect Analytics Streams and Big Data, e.g. kafka, Kinesis, kdb+)

http://ie.linkedin.com/in/prisconapoli

THE CHALLENGE

EMAIL SERVICE

"Begin at the beginning," the King said, very gravely,
"and go on till you come to the end: then stop".

1. Accepts the necessary information and sends emails
2. Provide an abstraction between two different email service providers
3. Quickly failover if a service goes down without affecting the customers

THOUGHTS

Would you tell me, please, which way I ought to go from here?
That depends a good deal on where you want to get to.

- Emails can be sent from any device, every time, everywhere
- The service should be accessible across Internet
- Manage a pool of mail service providers to guarantee availability
- Define the logic to pick up a mail provider for every incoming request
- Define a retry policy if a mail provider fails to serve a request
- Distribute the load when the number of request goes higher and higher

WHAT IF…

Curiouser and curiouser!

An email is lost:

- when this email entered the system?
- has it been discarded during the validation?
- was the queue down?
- what are the providers selected to serve this email?
- was a general network issue or just a provider outage?

The system is slow:

- what was the maximum time to delivery a mail?
- what is the average time spent in the queue? 95th/99th percentile?
- who is the slowest mail provider?
- can you show me success/failure rates?

There is a crash or an hardware failure:

- how persist data so any non-processed email is lost?
- how restart the whole processes like if nothing happened?

GOALS

Why, sometimes I've believed as many as six impossible things before breakfast.

availability: use standard Internet architecture and protocols
e.g. RESTful web service, HTTP/JSON

scalability: take advantage of additional computational and storage resources, development teams can work in parallel on different components

reliability: no data loss, define retry policy in case of failures

robustness: handling partial failures, quick failover, graceful degradation

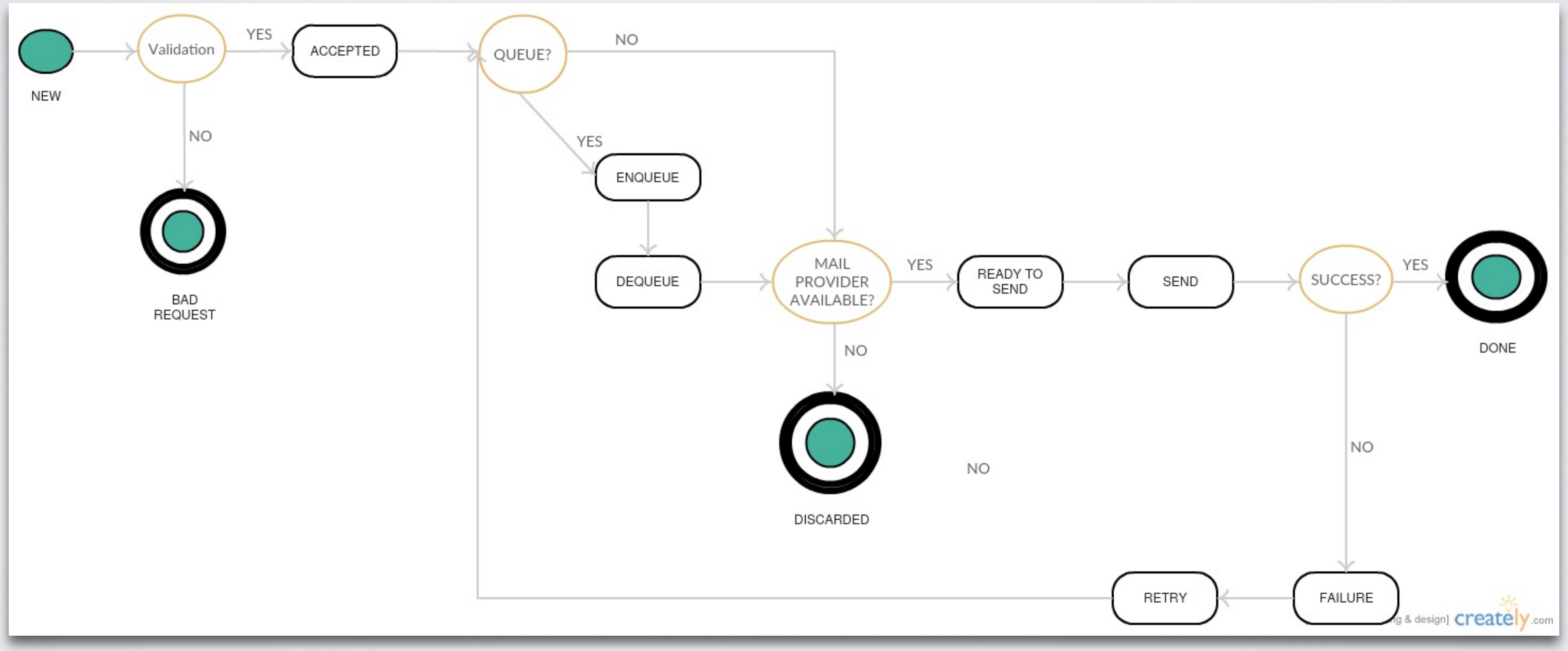
operations friendly: deploy and monitor without pain

security: firewall, DDoS attack, SSL/TLS and HTTP/2 termination, authentication, access control, secure API key

ARCHITECTURE

EMAIL MODEL

How puzzling all these changes are!
I'm never sure what I'm going to be, from one minute to another.



MAIL SERVICE

Well, I never heard it before, but it sounds uncommon nonsense.

```
class MailService(object):
    """ Abstract Class to model a Mail Service Provider """
    def name(self)
    def full_name(self)
    def send(self, mail, url_events=None)
    def post_process(self, response, mail, url_events=None)
    @abc.abstractmethod
    def response_details(self, response)
    @abc.abstractmethod
    def prepare_message(self, sender, subject, recipient, content)
    @abc.abstractmethod
    def send_message(self, message)
```

DATA MODEL

It's the oldest rule in the book.

Mail

field	type
id	int
sender	string
subjects	string
recipients	string
content	text

Event

field	type
id	int
created_at	long int
created_by	long int
event	string
mail	int
blob	string

DISPATCH AND RETRY POLICY

If you don't know where you are going any road can take you there.

For a new request, select an available mail providers and use it to send the email.

But…

- Who are the available mail providers in this moment?
- How to avoid selecting a mail provider twice for the same request?
- How to ignore a mail provider that no longer works?
- How to reduce the pressure on a mail provider in trouble?
- What happen if there are no mail providers at all?

DISPATCH AND RETRY POLICY

How long is forever? Sometimes, just one second.

Every request carries the information about the failed attempts, e.g. a list of strings with the service name

- **pro**: stateless dispatcher
- **cons**: extra space

New request: randomly select a mail provider

Failed request: use the failed attempts to randomly select a mail provider that has not been used before

Monitor for failures: wrap a mail provider with a circuit breaker object

ASYNC TASK QUEUE

I almost wish I hadn't gone down that rabbit-hole.

Send an email takes time (tcp socket, network connectivity, endpoint availability etc. etc.)

Asynchronous Task Queue

- Leave the application free to respond to client requests
- Send emails without blocking the main application

Celery + Redis

RESTful API

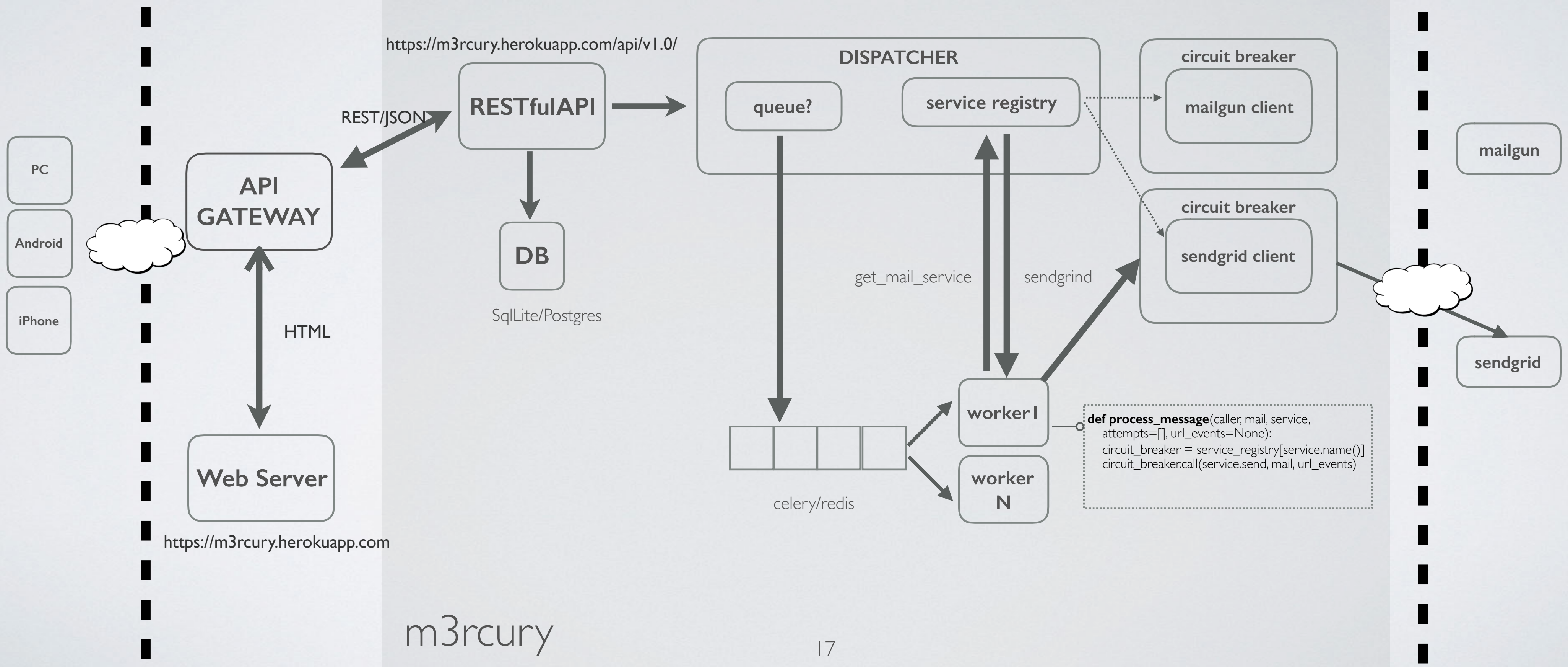
Take care of the sense, and the sounds will take care of themselves.

HTTP Method	URI	Action
GET	http[s]://[hostname]/api/v1.0/	Retrieve the API version and endpoints
GET	http[s]://[hostname]/api/v1.0/mails/	Retrieve the collection of all the email
POST	http[s]://[hostname]/api/v1.0/mails/	Create a new email
GET	http[s]://[hostname]/api/v1.0/mails/[mail_id]	Retrieve an email
GET	http[s]://[hostname]/api/v1.0/mails/[mail_id]/events/	Retrieve the collection of events by email
POST	http[s]://[hostname]/api/v1.0/mails/[mail_id]/events/	Create a new event
GET	http[s]://[hostname]/api/v1.0/mails/[mail_id]/events/[event_id]	Retrieve an event

```
api_1_0
├── __init__.py
├── decorators.py
├── errors.py
├── event
|   ├── __init__.py
|   ├── models.py
|   └── view.py
├── mail
|   ├── __init__.py
|   ├── models.py
|   └── views.py
├── mail_dispatcher
|   ├── __init__.py
|   ├── dispatcher.py
|   ├── exceptions.py
|   ├── queue.py
|   └── worker.py
├── mail_service
|   ├── __init__.py
|   ├── exceptions.py
|   ├── mail_service.py
|   ├── mailgun_service.py
|   └── sendgrid_service.py
└── utils.py
```

ARCHITECTURE

"Explain all that," said the Mock Turtle.



```
def process_message(caller, mail, service,
    attempts=[], url_events=None):
    circuit_breaker = service_registry[service.name()]
    circuit_breaker.call(service.send, mail, url_events)
```

DEMO

“No, no! The adventures first,” said the Gryphon in an impatient tone: “explanations take such a dreadful time”.

https://m3rcury.herokuapp.com

FUTURE WORK

How doth the little crocodile improve his shining tail,
and pour the waters of the Nile on every golden scale!

- service registry

HTTP Method	URI	Action
GET	http[s]://[hostname]/api/v1.0/dispatcher/providers/	Retrieve the collection of all the mail services
POST	http[s]://[hostname]/api/v1.0/dispatcher/providers/	Add a new mail service provider
PUT	http[s]://[hostname]/api/v1.0/dispatcher/providers/<id>	Update service registration by id
GET	http[s]://[hostname]/api/v1.0/dispatcher/providers/<id>	Get the an available mail service provider by id
DELETE	http[s]://[hostname]/api/v1.0/dispatcher/providers/<id>	Delete a mail service provider by id

- configuration management

HTTP Method	URI	Action
GET	http[s]://[hostname]/api/v1.0/configurations/	Retrieve the collection of all the configurations
POST	http[s]://[hostname]/api/v1.0/configurations/	Add a new configuration for a specific service
PUT	http[s]://[hostname]/api/v1.0/configurations<id>	Update configuration by id
GET	http[s]://[hostname]/api/v1.0/configurations/<id>	Get the configuration by id
DELETE	http[s]://[hostname]/api/v1.0/configurations/<id>	Delete configuration by id

FUTURE WORK

How doth the little busy bee improve each shining hour,
and gather honey all the day, from every opening flower!

- dynamic web pages
- performance monitoring, alerts
- event driven system
- auto-scaling

Q&A

Do you mean that you think you can find out the answer to it?

THANK YOU

But I don't want to go among mad people….
Oh, you can't help that, we're all mad here.
I'm mad. You're mad.