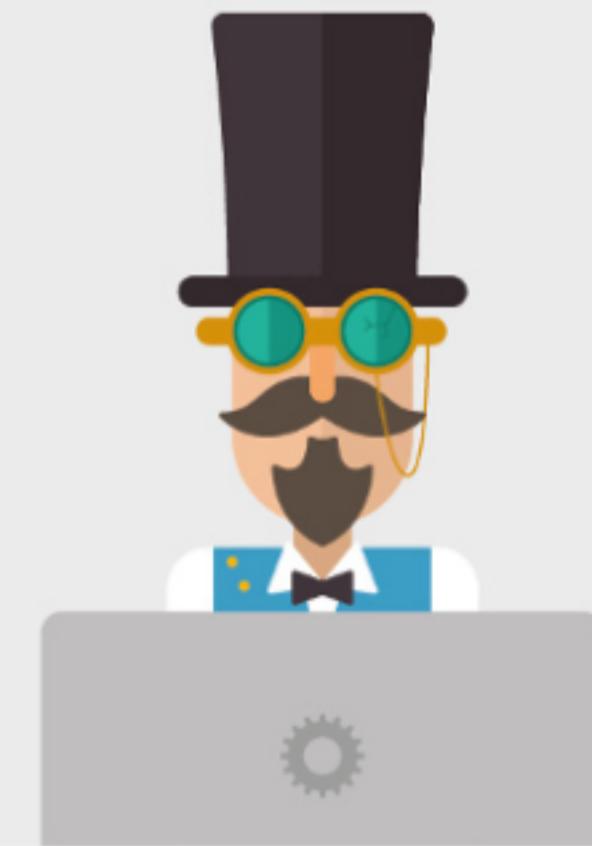


From spaghetti with no src/test to green CI and well-sleeping developers

Jacek Kunicki (@rucek)

Michał Matłoka (@mmatloka)



Outline

Outline

- overview

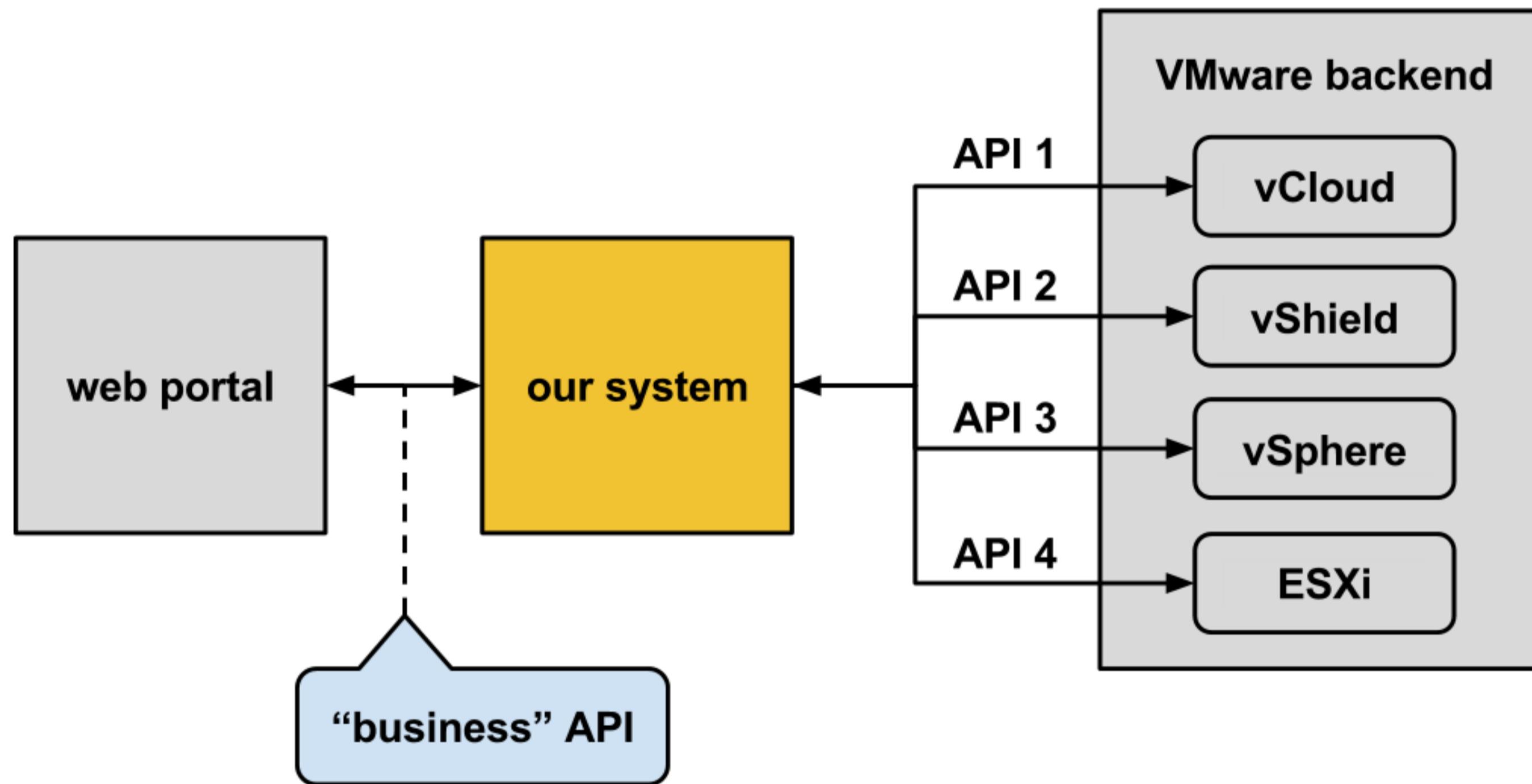
Outline

- overview
- initial situation

Outline

- overview
- initial situation
- our improvements

Overview



Initial situation

```
public String handleBadRequestException(BadRequestException e)
{
    GenericExceptionHandler.printStackTrace(e);
    JSONObject json = new JSONObject();
    json.put("message", exception.getMessage());
    json.put("cause" , exception.getCause());
    json.put("class" , exception.getClass().toString());
    logger.info(json.toJSONString());
    return json.toJSONString();
}
```



Spaghetti

- lots of copy-paste

Spaghetti

- lots of copy-paste
- large commented-out parts, inconsistent formatting

Spaghetti

- lots of copy-paste
- large commented-out parts, inconsistent formatting
- API calls not implemented or too buggy

Spaghetti

- Spring without DI

Spaghetti

- Spring without DI
- no input validation

Spaghetti

- Spring without DI
- no input validation
- every controller method returns a `String`

No tests failed...

No tests failed...

... but none of them were green
either

No tests failed...

... but none of them were green
either

```
$ ls src  
main
```



What we wanted to have
troubleshooting

Troubleshooting

- informative error messages

Troubleshooting

- informative error messages
- request tracking (async too)

Troubleshooting

- informative error messages
- request tracking (async too)
- verbose logging

Troubleshooting

- informative error messages
- request tracking (async too)
- verbose logging
- version information (almost) everywhere

Informative error messages

```
public enum ApiCallStatus {  
  
    ORGANIZATION_CREATED(  
        10100,  
        "Organization created successfully",  
        HttpStatus.CREATED  
    ),  
  
    // ...  
}
```

what we wanted to have / troubleshooting / informative error messages

Informative error messages

```
public enum ApiCallStatus {  
  
    ORGANIZATION_CREATED(  
        10100,  
        "Organization created successfully",  
        HttpStatus.CREATED  
    ),  
  
    // ...  
}
```

```
public class ApiCallException extends Exception {  
  
    private final ApiCallStatus apiCallStatus;  
  
    // ...  
}
```

what we wanted to have / troubleshooting / informative error messages

Informative error messages

```
@ResponseBody  
@ExceptionHandler  
public ResponseEntity<JsonResponse> handleApiCallException(ApiCallException e) {  
    logger.error("Handling exception", e);  
  
    final JsonResponse jsonResponse =  
        JsonResponse.create().forApiCallException(e);  
  
    return new ResponseEntity<>(  
        jsonResponse,  
        e.status().httpStatus().orElse(HttpStatus.INTERNAL_SERVER_ERROR)  
    );  
}
```

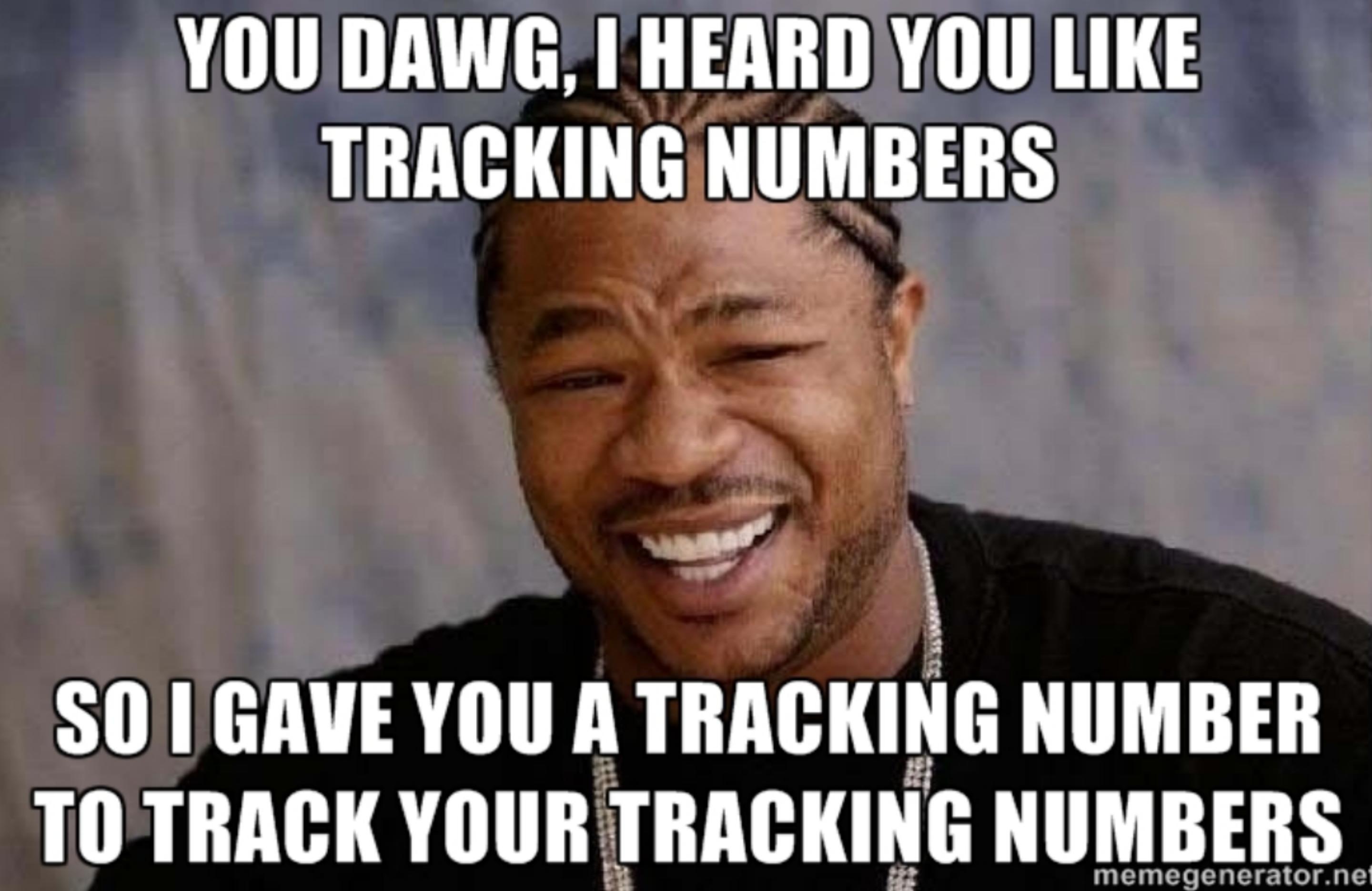
what we wanted to have / troubleshooting / informative error messages

Informative error messages

```
@ResponseBody  
@ExceptionHandler  
public ResponseEntity<JsonResponse> handleApiCallException(ApiCallException e) {  
    logger.error("Handling exception", e);  
  
    final JsonResponse jsonResponse =  
        JsonResponse.create().forApiCallException(e);  
  
    return new ResponseEntity<>(  
        jsonResponse,  
        e.status().httpStatus().orElse(HttpStatus.INTERNAL_SERVER_ERROR)  
    );  
}
```

```
{  
    "message": "Exception message",  
    "class": "com.example.AnException",  
    "cause": "java.io.IOException: Stream closed",  
    "statusCode": 42,  
    "statusMessage": "Something went wrong"  
}
```

what we wanted to have / troubleshooting / informative error messages

A photograph of Dwayne "The Rock" Johnson laughing heartily. He has short brown hair and is wearing a dark, zippered hoodie. The background is a soft-focus indoor setting.

**YOU DAWG, I HEARD YOU LIKE
TRACKING NUMBERS**

**SO I GAVE YOU A TRACKING NUMBER
TO TRACK YOUR TRACKING NUMBERS**

Request tracking evolution

what we wanted to have / troubleshooting / request tracking

Request tracking evolution

- each controller method

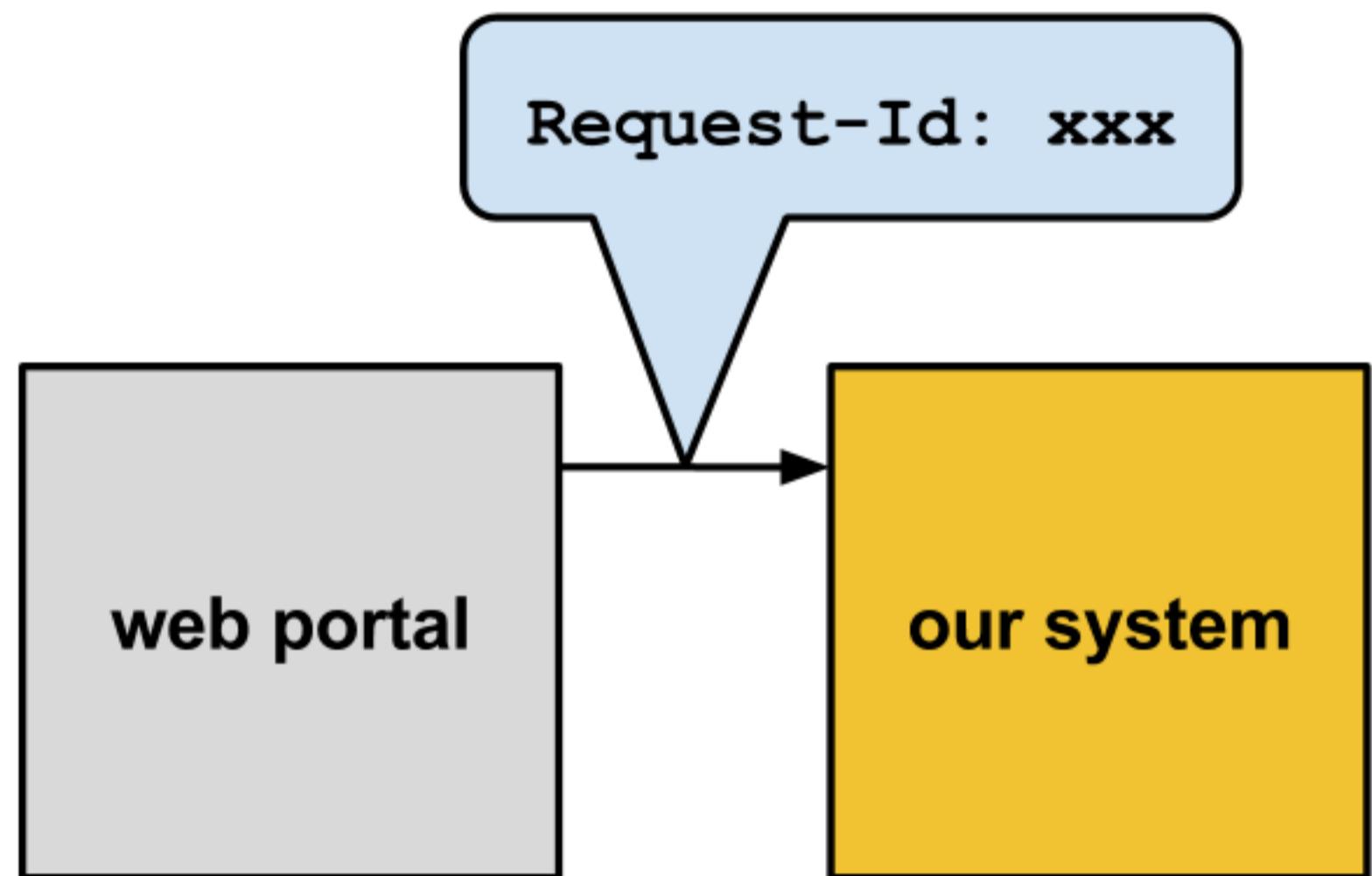
Request tracking evolution

- each controller method
- servlet filter

Request tracking evolution

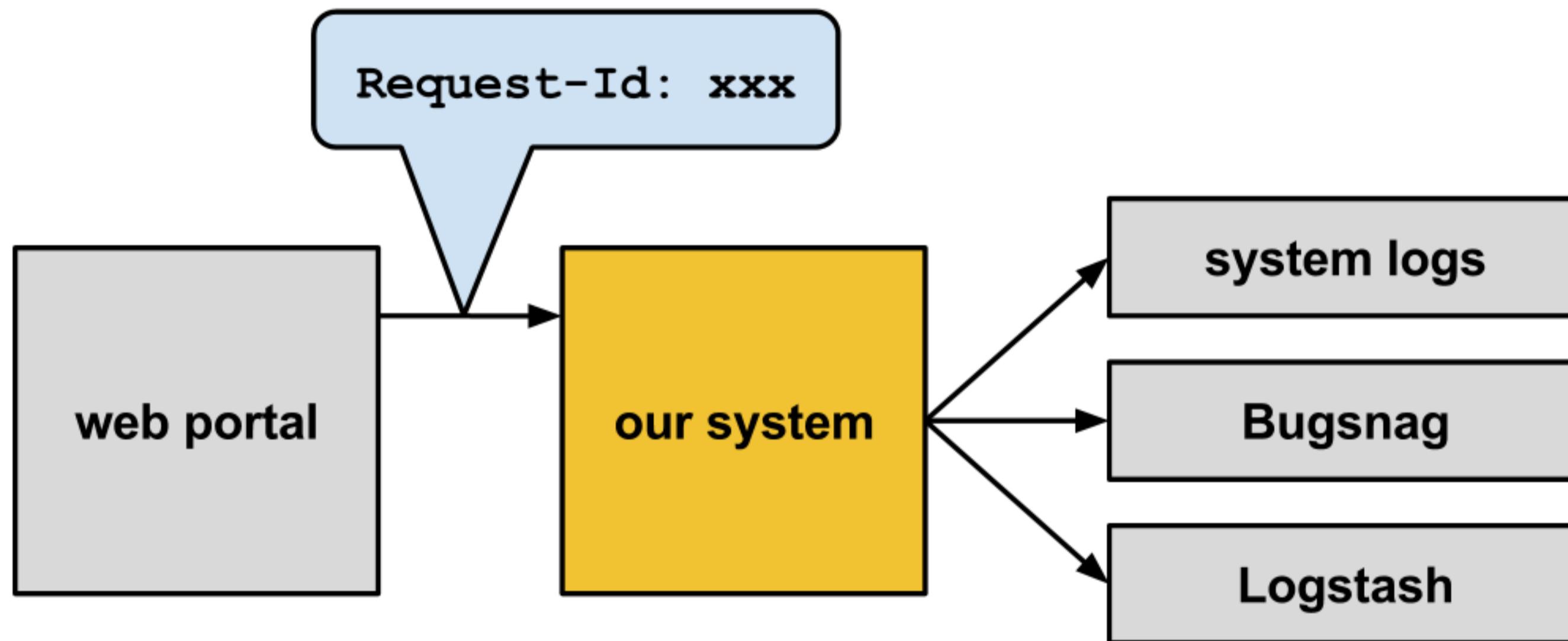
- each controller method
- servlet filter
- no more passing to other layers, no explicit returning

Cross-system request ID



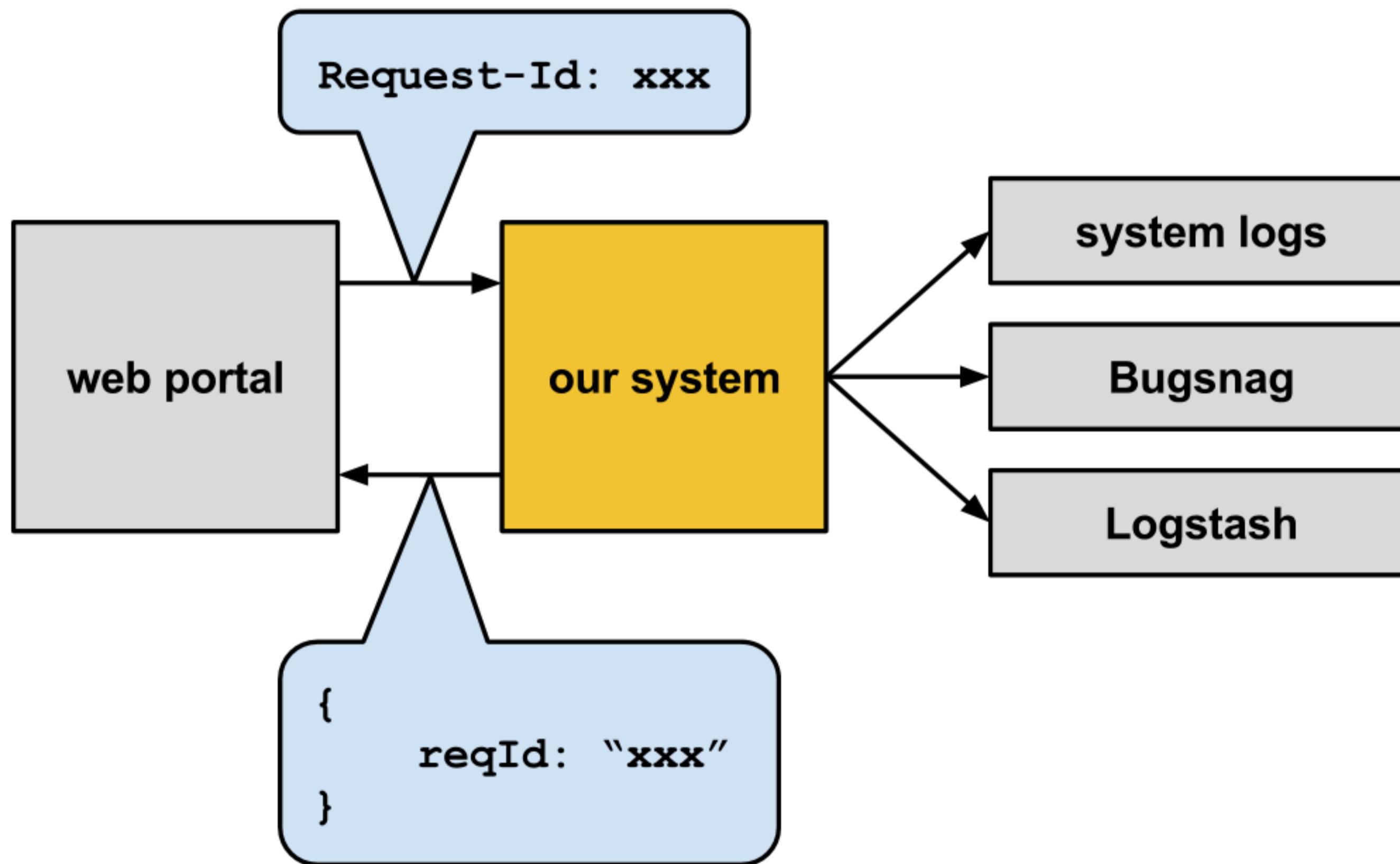
what we wanted to have / troubleshooting / request tracking

Cross-system request ID



what we wanted to have / troubleshooting / request tracking

Cross-system request ID



what we wanted to have / troubleshooting / request tracking

Bugsnag

Search errors 

81 - **java.net.NoRouteToHostException** · PlainSocketImpl.java:-2

EVENTS USERS

No route to host

History 2w 1d P/  PlainSocketImpl.java:-2 NEWEST NEXT 2015-02-26 · 04:45:02 · UTC PREV OLDEST

not seen in the last 14 days

● FIRST SEEN ● LAST SEEN

P FEB 26 P FEB 26

S DEC 18 S DEC 18

STACKTRACE REQUEST HOST ORIGINAL STACKTRACE EVENTS NEW TAB

id 6b7348b5-af18-48e0-96ce-f75cc67a0da6

url 127.0.0.1 DELETE /api/vm/vm-d6fb5e6b-5db3-4f30-ad7a-8818ff52c887

Logstash Search

QUERY ▶

● *vbroker

FILTERING ◀

ALL EVENTS



0 to 27 of 27 available for paging

View: [Table](#) / [JSON](#) / [Raw](#)

Field	Action	Value
@environment	Q Ø #	production
@fields.severity	Q Ø #	info
@message	Q Ø #	POST /api/stats
@source	Q Ø #	syslog://
@source_host	Q Ø #	sc1
@timestamp	Q Ø #	2015-04-13T00:30:19.400+00:00
_index	Q Ø #	logstash-2015.04.13
host	Q Ø #	10.158.41.29
javafile	Q Ø #	c._.c.b.filter.RequestLoggingFilter
javafile_line	Q Ø #	165
logdate	Q Ø #	2015-04-12 17:30:17,944
loglevel	Q Ø #	info
request_id	Q Ø #	a543885b-d421-4b1e-bdc1-2b22927009d2
request_type	Q Ø #	R
tags	Q Ø #	json,udp,@message_exists,pre_formated,vbroker_match,_grokparsefailure
thread	Q Ø #	qtp477376212-48786
type	Q Ø #	json

MDC

what we wanted to have / troubleshooting / request tracking

MDC

- Mapped Diagnostic Context, a part of the logging framework

what we wanted to have / troubleshooting / request tracking

MDC

- Mapped Diagnostic Context, a part of the logging framework
- per-thread, inherited

what we wanted to have / troubleshooting / request tracking

MDC

- Mapped Diagnostic Context, a part of the logging framework
- per-thread, inherited

```
MDC.put("requestId", requestId);
```

MDC

- Mapped Diagnostic Context, a part of the logging framework
- per-thread, inherited

```
MDC.put("requestId", requestId);
```

```
encoder(PatternLayoutEncoder) {  
    pattern = "%date [%thread] [%X{requestId}] %-5level %logger{36}:%line - %msg%n"  
}
```

Verbose logging

```
2015-03-31 13:15:31,987 [qtp1842864232-132]
[R a543885b-d421-4b1e-bdc1-2b22927009d2]
INFO c.c.b.filter.RequestLoggingFilter:159
Request: 127.0.0.1 GET /api/task/req-2486d57f-6657-4efc-8a18-86eef055b8a6

2015-03-31 13:15:31,988 [qtp1842864232-132]
[R a543885b-d421-4b1e-bdc1-2b22927009d2]
INFO c.c.b.v.Magic:19
Cooking spaghetti

2015-03-31 13:15:31,989 [qtp1842864232-132]
[R a543885b-d421-4b1e-bdc1-2b22927009d2]
INFO c.c.b.filter.RequestLoggingFilter:134
Response body:
{"data": {"taskId": "req-2486d57f-6657-4efc-8a18-86eef055b8a6",
"taskState": "RUNNING"}, "statusMessage": "Task status retrieved",
"statusCode": 33100, "reqId": "a543885b-d421-4b1e-bdc1-2b22927009d2"}
```

what we wanted to have / troubleshooting / verbose logging

Version information

what we wanted to have / troubleshooting / version information

Version information

- Maven git plugin, Gradle snapshot plugin

Version information

- Maven git plugin, Gradle snapshot plugin
- version number included in startup logs and in Bugsnag error reports (including last commit info)

What we wanted to have
easier development

Easier development

what we wanted to have / easier development

Easier development

- as few boilerplate as possible

Easier development

- as few boilerplate as possible
- integration testing with a real backend

Easier development

- as few boilerplate as possible
- integration testing with a real backend
- SwaggerUI for quick manual testing without the portal part

Less boilerplate - goal

- have request id and response status in the response
- be able to specify http status
- don't need to write too much repeating code

what we wanted to have / easier development / less boilerplate

Less boilerplate - goal

- have request id and response status in the response
- be able to specify http status
- don't need to write too much repeating code

```
{  
  "data": [  
    {  
      "id": "...",  
      ...  
    },  
    ...  
  ],  
  "statusMessage": "Some types of objects were retrieved successfully",  
  "statusCode": 10010,  
  "reqId": "9f03c326-13dc-477c-acbb-7e13c1661d02"  
}
```

what we wanted to have / easier development / less boilerplate

Less boilerplate

```
// annotations...
public String someOperation(Map<String, Object> body) {
    ...
}
```

what we wanted to have / easier development / less boilerplate

Less boilerplate

```
// annotations...
public String someOperation(Map<String, Object> body) {
    ...
}
```

```
// annotations...
public ResponseEntity<SomeType> someOperation(Map<String, Object> body) {
    ...
    return new ResponseEntity<SomeType>(someType, HttpStatus.CREATED);
}
```

what we wanted to have / easier development / less boilerplate

Less boilerplate

```
// annotations...
public String someOperation(Map<String, Object> body) {
    ...
}
```

```
// annotations...
public ResponseEntity<SomeType> someOperation(Map<String, Object> body) {
    ...
    return new ResponseEntity<SomeType>(someType, HttpStatus.CREATED);
}
```

```
// annotations...
public JsonResponse someOperation(Map<String, Object> body) {
    ...
    return JsonResponse.create().forApiCallStatus(...).withData(...)
        .withTaskIdFromRequest();
}
```

what we wanted to have / easier development / less boilerplate

Less boilerplate

```
// annotations...
public String someOperation(Map<String, Object> body) {
    ...
}
```

```
// annotations...
public ResponseEntity<SomeType> someOperation(Map<String, Object> body) {
    ...
    return new ResponseEntity<SomeType>(someType, HttpStatus.CREATED);
}
```

```
// annotations...
public JsonResponse someOperation(Map<String, Object> body) {
    ...
    return JsonResponse.create().forApiCallStatus(...).withData(...)
        .withTaskIdFromRequest();
}
```

```
// annotations with specified api call status (and http status)
public List<SomeDto> someOperation(SomeObject someObject) {
    return someManager.doSth(someObject);
}
```

what we wanted to have / easier development / less boilerplate

Less boilerplate

```
@ApiCall(succeedsWith = ApiCallStatus.AD_RAN_UPDATE_GROUP_SCRIPT)
@RequestMapping(value = "/{identity}",
                 method = RequestMethod.PUT,
                 produces = JsonResponse.CONTENT_TYPE)
@ResponseStatus(HttpStatus.OK)
```

what we wanted to have / easier development / less boilerplate

Less boilerplate

```
@ApiCall(succeedsWith = ApiCallStatus.AD_RAN_UPDATE_GROUP_SCRIPT)
@RequestMapping(value = "/{identity}",
                 method = RequestMethod.PUT,
                 produces = JsonResponse.CONTENT_TYPE)
@ResponseStatus(HttpStatus.OK)
```

```
@Put(value = "/{identity}",
      succeedsWith = ApiCallStatus.AD_RAN_UPDATE_GROUP_SCRIPT)
```

what we wanted to have / easier development / less boilerplate

Less boilerplate

```
@Get(succeedsWith = ApiCallStatus.SOME_STATUS)
public List<SomeDto> someOperation(...) throws SomeException {
    ...
}
```

```
{
    "data": [
        {
            "id": "...",
            ...
        },
        ...
    ],
    "statusMessage": "Some types of objects were retrieved successfully",
    "statusCode": 10010,
    "reqId": "9f03c326-13dc-477c-acbb-7e13c1661d02"
}
```

what we wanted to have / easier development / less boilerplate

Less boilerplate - solution

- AspectJ

what we wanted to have / easier development / less boilerplate

Less boilerplate - solution

- AspectJ
- RequestResponseBodyMethodProcessor

what we wanted to have / easier development / less boilerplate

Less boilerplate - solution

- AspectJ
- RequestResponseBodyMethodProcessor
- RequestMappingHandlerMapping

what we wanted to have / easier development / less boilerplate

Less boilerplate - validation

```
// annotations...
public String someOperation(Map<String, Object> body) {
    ...
}
```

what we wanted to have / easier development / less boilerplate

Less boilerplate - validation

```
// annotations...
public String someOperation(Map<String, Object> body) {
    ...
}
```

```
// annotations...
public SomeDto someOperation(@RequestBody @Valid SomeCommand body) {
    ...
}

class SomeCommand {

    @NotNull
    private String name;
    ...
}
```

what we wanted to have / easier development / less boilerplate

Less boilerplate - async

```
@Async  
@ApiCall(succeedsWith = ApiCallStatus.ORGANIZATION_DELETED,  
         failsWith = ApiCallStatus.ORGANIZATION_DELETE_FAILED)  
public Future<SomeObject> executeOnOtherThread() {  
    ...  
}
```

=

```
... implements AsyncTaskExecutor {  
  
    @Override  
    public <T> Future<T> submit(Callable<T> task) {  
        ...  
    }  
    ...  
}
```

what we wanted to have / easier development / less boilerplate

Integration tests

what we wanted to have / easier development / integration tests

Integration tests

- Spring Boot - `@IntegrationTest`

what we wanted to have / easier development / integration tests

Integration tests

- Spring Boot - `@IntegrationTest`
- Spock

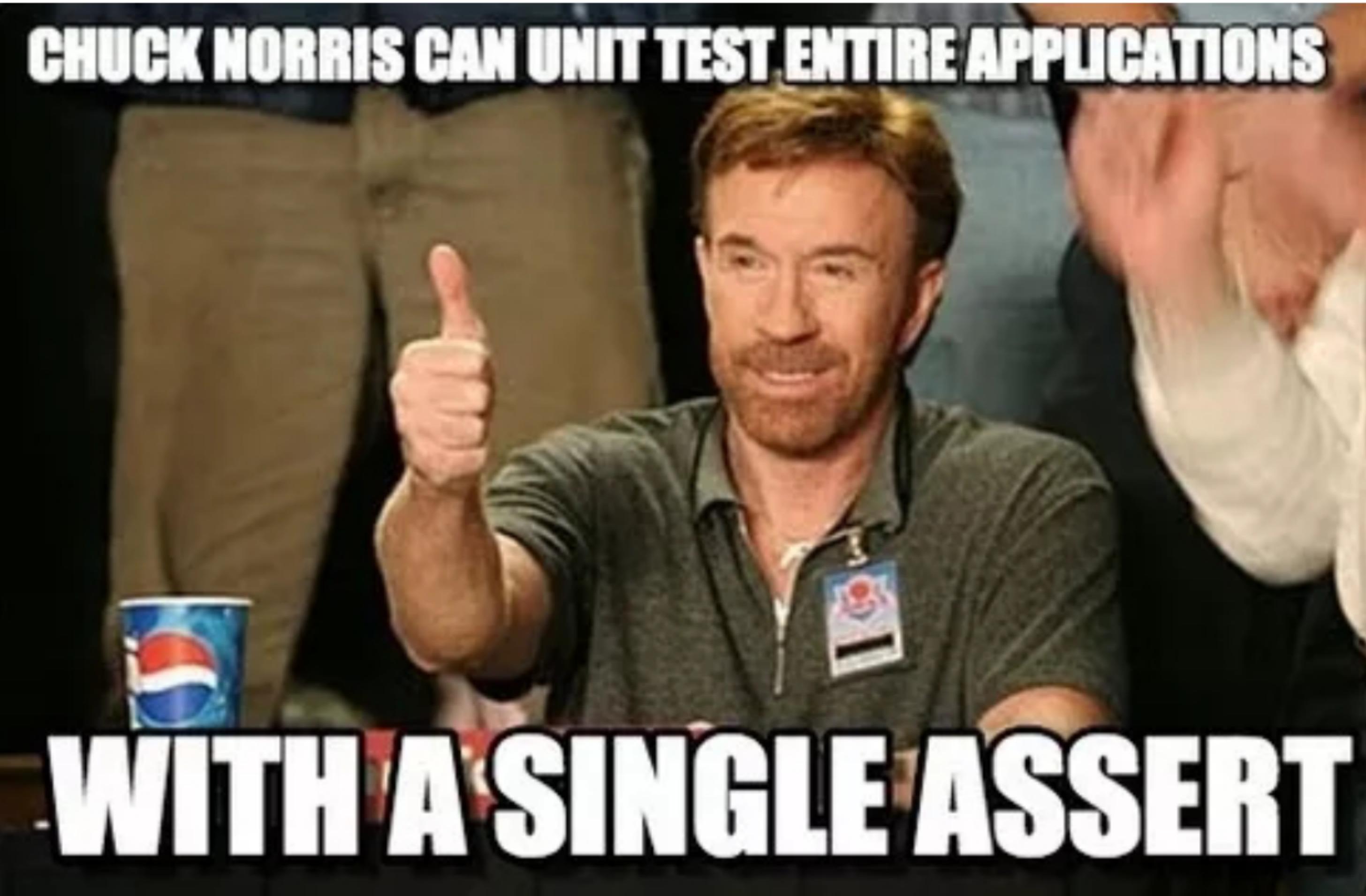
what we wanted to have / easier development / integration tests

Integration tests

- Spring Boot - `@IntegrationTest`
- Spock
- rest-assured

what we wanted to have / easier development / integration tests

CHUCK NORRIS CAN UNIT TEST ENTIRE APPLICATIONS



WITH A SINGLE ASSERT

Integration tests - Groovy goodness

```
def 'should not delete a nonexistent organization'() {  
    when:  
        def taskId = when().delete('/organization/fooBar')  
            .then().extract().path('taskId')  
  
    then:  
        awaitTaskState(taskId, TaskState.FAILED)  
}
```

what we wanted to have / easier development/integration tests

Integration tests - Groovy goodness

```
def 'should not delete a nonexistent organization'() {  
    when:  
        def taskId = when().delete('/organization/fooBar')  
            .then().extract().path('taskId')  
  
    then:  
        awaitTaskState(taskId, TaskState.FAILED)  
}
```

VS.

```
def 'should not delete a nonexistent organization'() {  
    when:  
        def taskId = whenOrganizationIsDeleted('fooBar').then().extractTaskId()  
  
    then:  
        awaitTaskState(taskId, TaskState.FAILED)  
}
```

what we wanted to have / easier development/integration tests

POST

/rest/users/

Authenticate user

Response Class

[Model](#) | [Model Schema](#)

```
{  
    "id": "",  
    "login": "",  
    "email": "",  
    "token": ""  
}
```

Response Content Type [application/json](#)

Parameters

Parameter	Value	Description	Parameter Type	Data Type
body	(required)	Authentication data	body	Model Model Schema

```
{  
    "login": "",  
    "password": "",  
    "rememberMe": false  
}
```

Click to set as parameter value

Response Messages

HTTP Status Code	Reason	Response Model
200	OK	
401	Invalid login and/or password	

[Try it out!](#)

what we wanted to have / easier development / Swagger UI

What we wanted to have
workflow

Workflow

- CI & automation

what we wanted to have / workflow

Workflow

- CI & automation
- GitFlow

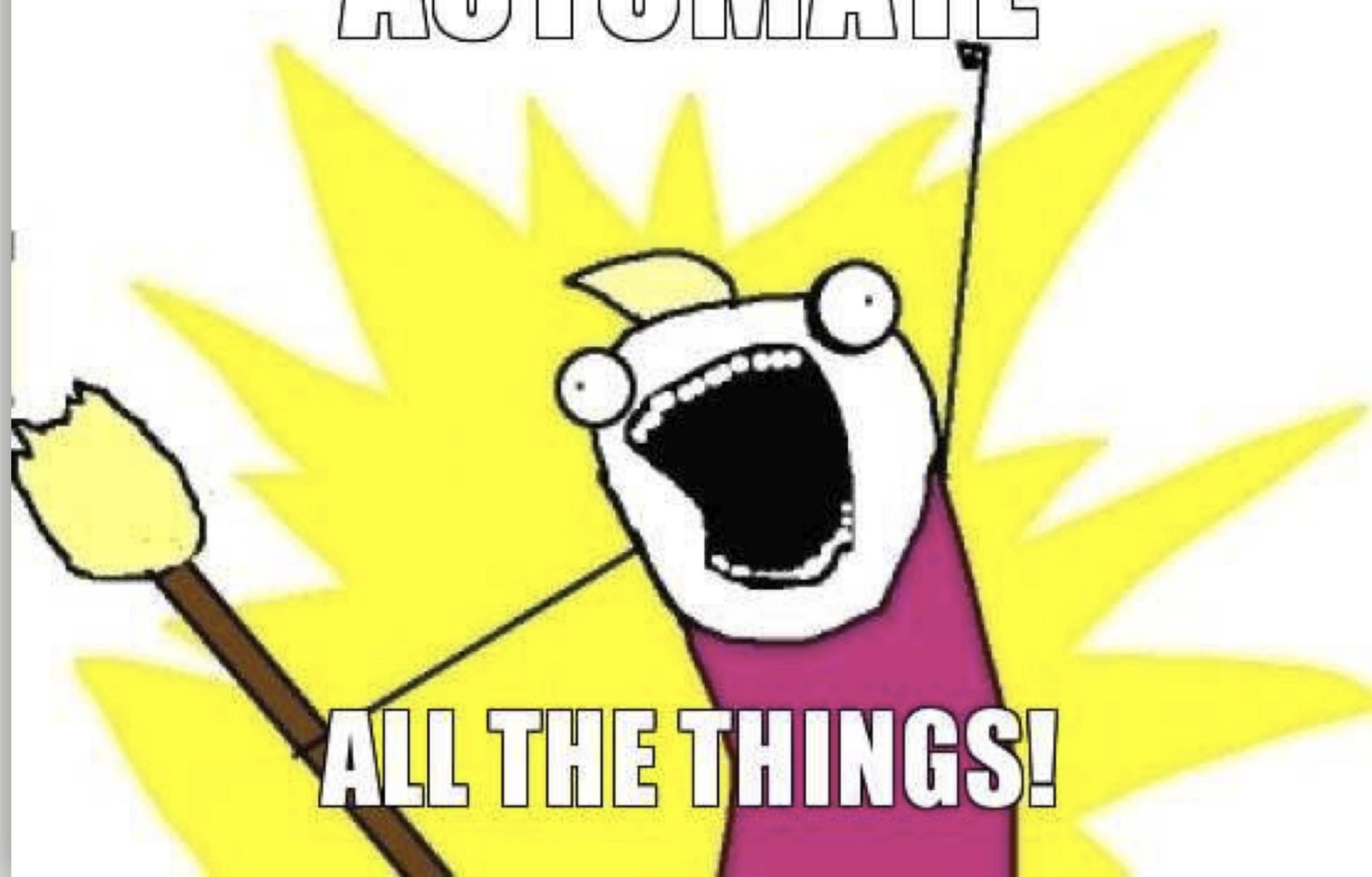
what we wanted to have / workflow

Workflow

- CI & automation
- GitFlow
- semantic versioning

what we wanted to have / workflow

AUTOMATE



ALL THE THINGS!

CI & automation

- CI build on every commit (also on branches)

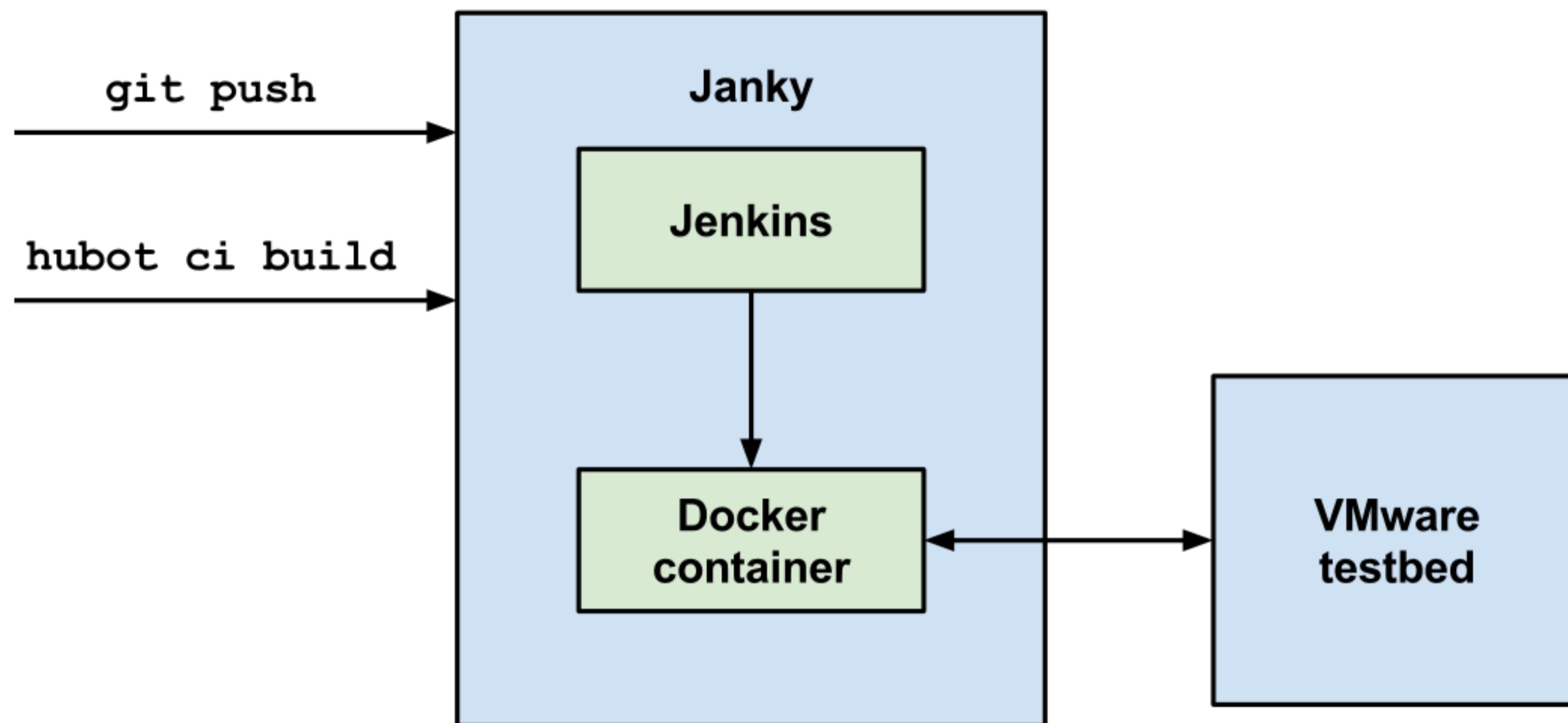
CI & automation

- CI build on every commit (also on branches)
- Chef-managed environments

CI & automation

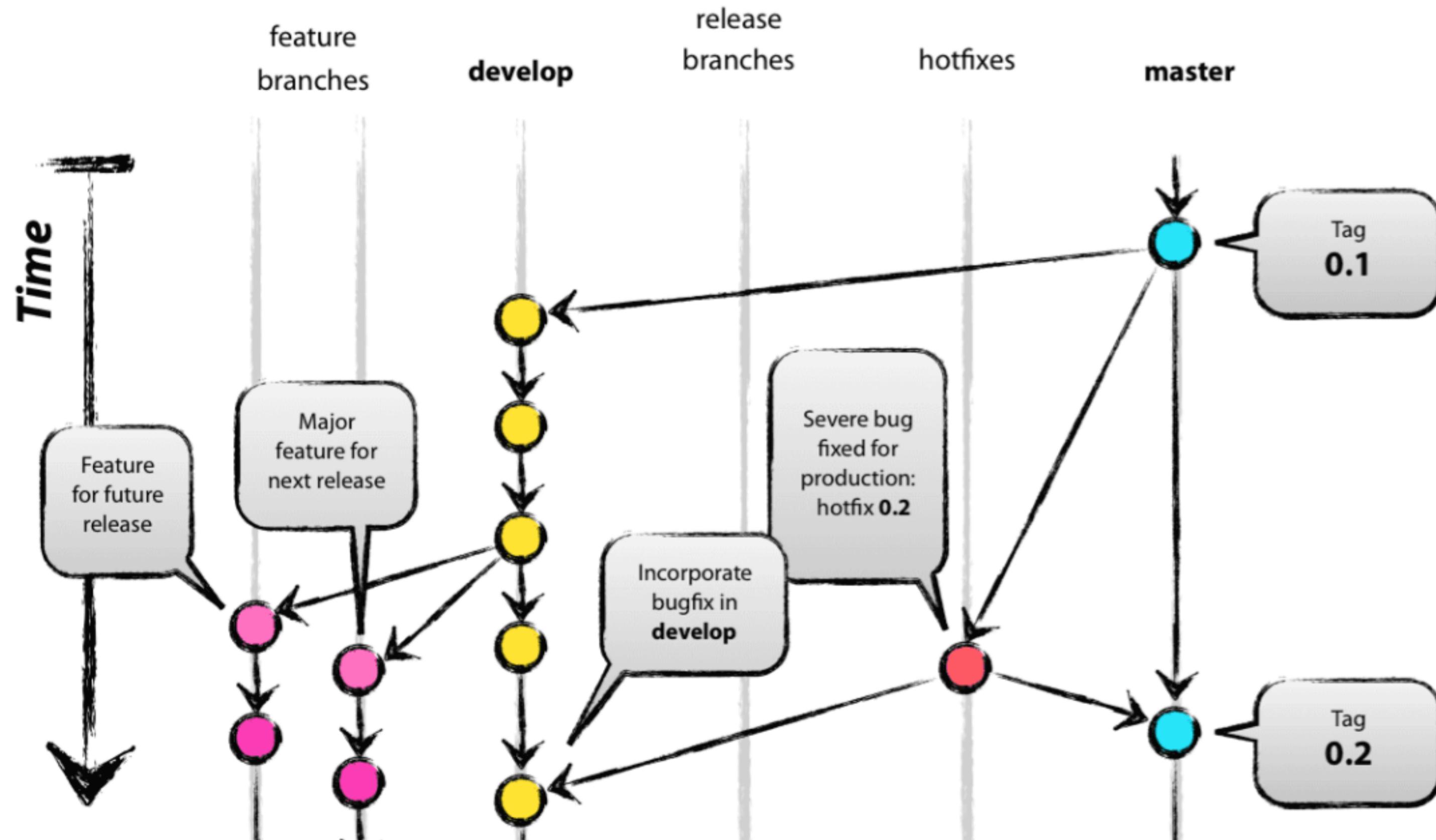
- CI build on every commit (also on branches)
- Chef-managed environments
- environment-related tags in Git

CI & automation

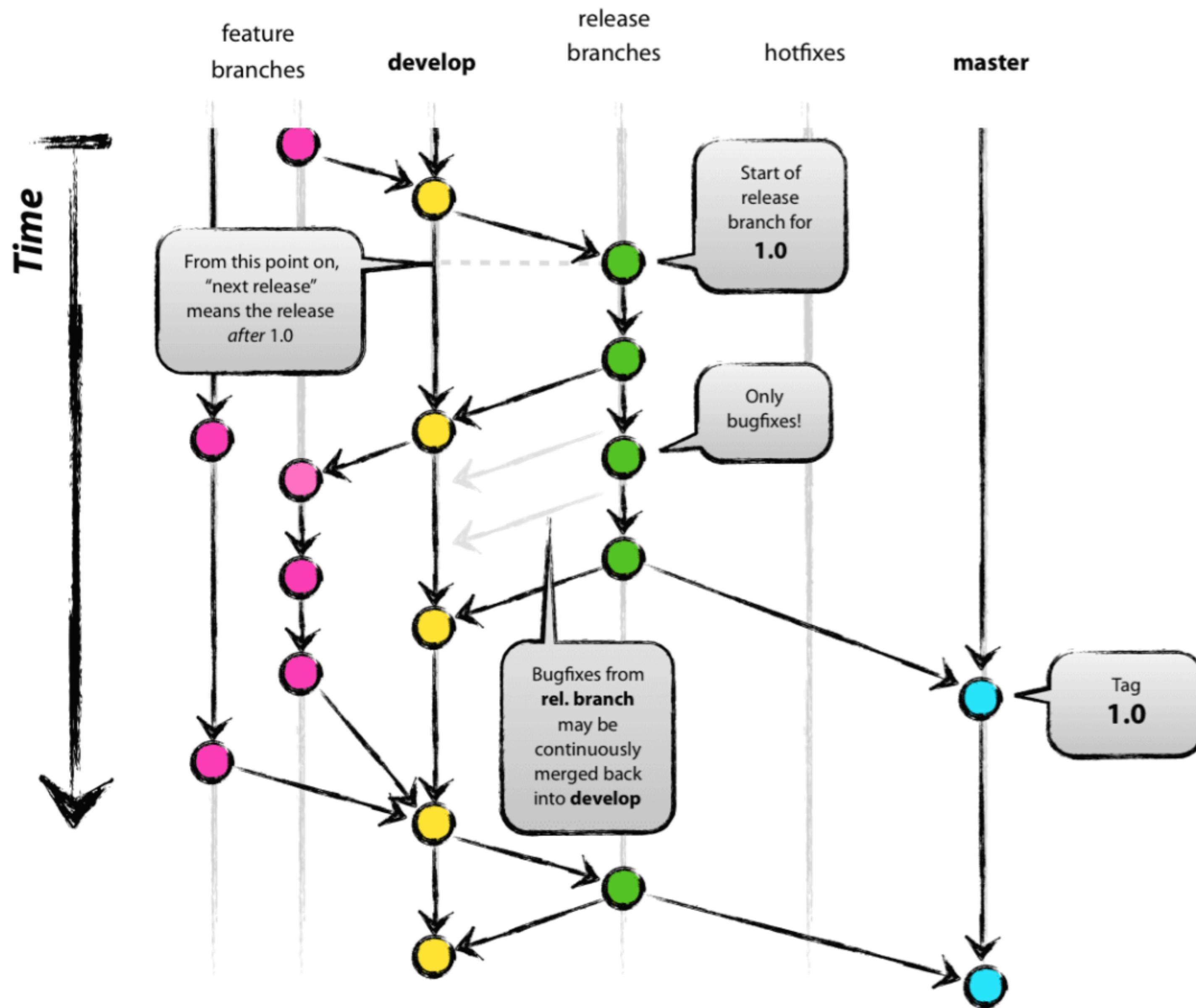


what we wanted to have / workflow / CI and automation

GitFlow



what we wanted to have / workflow / GitFlow



what we wanted to have / workflow / GitFlow

Semantic versioning

3.2.8
(major.minor.patch)

Semantic versioning

3.2.8
(major.minor.patch)

- patch = bug fix, backwards-compatible

Semantic versioning

3.2.8
(major.minor.patch)

- patch = bug fix, backwards-compatible
- minor = new feature, backwards-compatible

Semantic versioning

3.2.8
(major.minor.patch)

- patch = bug fix, backwards-compatible
- minor = new feature, backwards-compatible
- major = new feature, backwards-incompatible

**THIS DEPARTMENT
HAS WORKED**

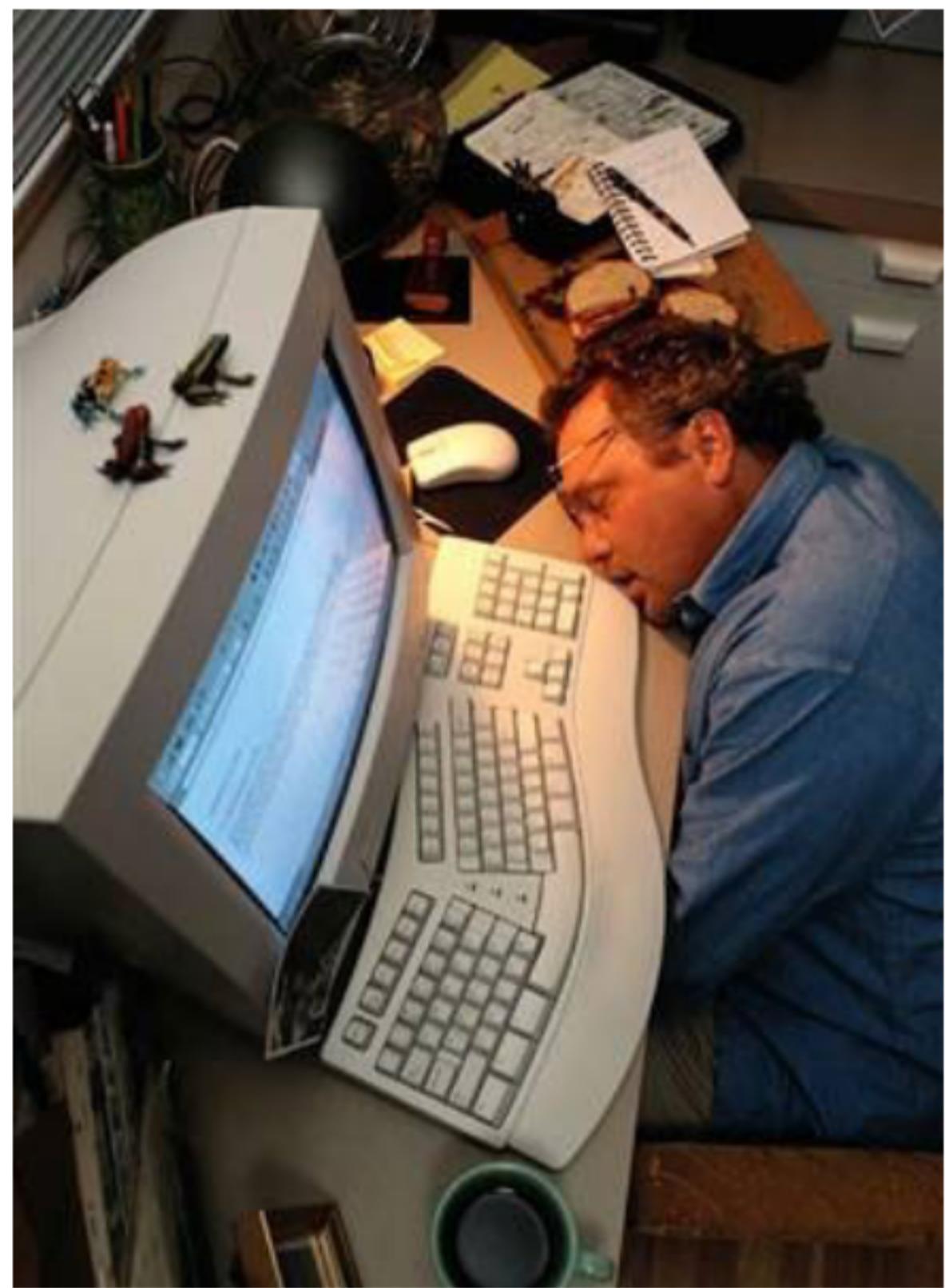
297 □ DAYS

**WITHOUT A LOST
TIME ACCIDENT**

**THE BEST PREVIOUS
RECORD WAS**

0 □ DAYS

**DO YOUR PART!
HELP MAKE
A NEW RECORD**



Summing up

Summing up

- stay DRY

Summing up

- stay DRY
- log and track everything that is important for debugging

Summing up

- stay DRY
- log and track everything that is important for debugging
- organize your workflow (automate!!!)

Summing up

- stay DRY
- log and track everything that is important for debugging
- organize your workflow (automate!!!)
- don't be afraid to experiment

Thank you, Q&A?

@rucek

@mmatloka

Slides:

<http://www.slideshare.net/SoftwareMill/p-46980945>

Image credits:

http://fc01.deviantart.net/fs45/f/2009/087/f/1/Mario_and_Luigi_by_LuigiL.jpg
<https://momeefriendsls.files.wordpress.com/2013/04/mommy-blog-1-0033.jpg>
<https://www.cheffio.com/wp-content/uploads/2012/02/automate-all-the-things.png>
<https://www.laborlawcenter.com/images/Product/large/RIS-MSR124AL-R695.jpg>
<http://www.vanitatis.elconfidencial.com/cache/2008/03/18/95trabajoinferior.jpg>
<http://qph.is.quoracdn.net/main-qimg-4a4f77e148fb05d7c5de2c00ed7644cb>
<http://cdn.meme.am/instances/1024x/54442655.jpg>
<http://nvie.com/posts/a-successful-git-branching-model/>