Developer, consultant, trainer, speaker

@michieltcs
Large aging monolith
Large aging monolith

Generates income
Large aging monolith

Generates income

Slow & complex
Large aging monolith

Technical debt

Slow & complex

Generates income
Manual releases
Manual releases

Fragile tests
Manual releases

Low velocity

Frequent issues

Fragile tests
GOALS
GOALS

1. REDUCE ISSUES
GOALS

1. REDUCE ISSUES

2. REDUCE LEAD TIME
GOALS

1. REDUCE ISSUES
2. REDUCE LEAD TIME
3. INCREASE PRODUCTIVITY
GOALS

1. REDUCE ISSUES
2. REDUCE LEAD TIME
3. INCREASE PRODUCTIVITY
4. INCREASE MOTIVATION
REFACTOR? REBUILD?
APPROACH

- API first
- Services per domain object (job, jobseeker, ...)
- Migrate individual pages
ORIGINAL MONOLITH
RewriteEngine On

# Serve feature from new service for internal network
RewriteCond expr " %{HTTP:X-FORWARDED-FOR} -ipmatch '192.168.0.0/24'"
RewriteRule ^/feature/(.*)$ ${NEW_SERVICE_URL}/$1 [P,L]

# Proxy everything else to legacy application
RewriteRule ^/(.*) ${LEGACY_URL}/$1 [P]
ProxyPassReverse / ${LEGACY_URL}/
ARCHITECTURE

FRONTENDS ARE SERVICES
ARCHITECTURE

FRONTENDS ARE SERVICES

SERVICES BEHIND LOAD BALancers
ARCHITECTURE

FRONTENDS ARE SERVICES

SERVICES BEHIND LOAD BALANCERS

ACCESS LEGACY DB’S
ARCHITECTURE

- Frontends are services
- Services behind load balancers
- Access legacy DB’s
- Services in containers
PROCESS
CONTINUOUS EVERYTHING
CD?
CONTINUOUS INTEGRATION

DEV → BUILD / TEST
CONTINUOUS DELIVERY

DEV → BUILD / TEST → ACCEPTANCE → PRODUCTION
CONTINUOUS DEPLOYMENT

DEV → BUILD / TEST → STAGING / ACCEPTANCE → PRODUCTION
SMALL STEPS
SMALL STEPS

EARLY FEEDBACK
SMALL STEPS
EARLY FEEDBACK
REDUCE TIME TO RECOVER
SMALL STEPS

EARLY FEEDBACK

REDUCE TIME TO RECOVER

EXPERIMENTS!
200x more frequent deployments
24x faster recovery from failures
3x lower change failure rate
2,555x shorter lead times
EVERY COMMIT GOES TO PRODUCTION
ONLY COMMIT TO MASTER
NO BRANCHES
NO BRANCHES

REALLY.
PAIR PROGRAMMING
FOCUS ON VALUE
FEATURE TOGGLES, A/B TESTS
100% CODE COVERAGE *
DEVOPS
MONITORING
BUILD PIPELINE
AUTOMATE REPEATABLE THINGS
CONTINUOUS TESTING
DEFENSE IN DEPTH

UNIT TESTS → INTEGRATION TESTS → ACCEPTANCE TESTS → UI TESTS
@Test
public void jobCannotBeFound() {
    when(jobRepository.getById(EXPECTED_JOB_ID))
        .thenReturn(null);

    JobService jobService = new JobService(jobRepository);

    assertNull(jobService.getById(EXPECTED_JOB_ID));
    verify(jobRepository).getById(EXPECTED_JOB_ID);
}
@Test
public void shouldFindJob() {
    expectedJob = loadFixture('active_job.yml');
    actualJob = repository.getById(expectedJob.getId());

    assertThat(actualJob, isA(Job.class));
    assertEquals(expectedJob.getId(), actualJob.getId());
}
Scenario: Link to related job

Given a job exists
And there are related jobs available
When that job is viewed
Then a list of related jobs is shown
And each related job links to the detail page of the related job
DEFENSE IN DEPTH

UNIT TESTS ➔ INTEGRATION TESTS ➔ ACCEPTANCE TESTS ➔ UI TESTS

@michieltcs
DEFENSE IN DEPTH

UNIT TESTS

INTEGRATION TESTS

ACCEPTANCE TESTS

UI

@michieltcs
CONTINUOUS TESTING

- Unit Tests
- Integration Tests
- Acceptance Tests
- UI Tests
- Smoke Tests

Exploratory testing

Cost

Monitoring

Speed

@michieltcs
node {
    stage('Run tests') {
        sh "phpunit"
        sh "behat"
    }

    stage('Build docker image') {
        sh "docker build -t jobservice:${env.BUILD_NUMBER} ." 
        sh "docker push jobservice:${env.BUILD_NUMBER}" 
    }

    stage('Deploy staging') {
        sh "ansible-playbook -e BUILD=${env.BUILD_NUMBER} -i staging deploy.yml"
    }

    stage('Deploy production') {
        sh "ansible-playbook -e BUILD=${env.BUILD_NUMBER} -i prod deploy.yml"
    }
}
DEPLOYING: ROLLING UPDATE

1. PULL IMAGE
2. START NEW CONTAINER
3. WAIT FOR PORT
4. SMOKE TESTS / HEALTH CHECKS
5. ADD NEW CONTAINER TO LB
6. REMOVE OLD CONTAINER FROM LB
7. STOP OLD CONTAINER

@michielcts
DEPLOYING

PULL IMAGE

START NEW CONTAINER

WAIT FOR PORT

SMOKE TESTS / HEALTH CHECKS

ADD NEW CONTAINER TO LB

REMOVE OLD CONTAINER FROM LB

STOP OLD CONTAINER

`docker pull`
DEPLOYING

1. Pull Image
2. Start New Container
3. docker run
4. Wait for Port
5. Smoke Tests / Health Checks
6. Add New Container to LB
7. Remove Old Container from LB
8. Stop Old Container
DEPLOYING

1. PULL IMAGE
2. START NEW CONTAINER
3. WAIT FOR PORT
4. SMOKE TESTS / HEALTH CHECKS
5. ADD NEW CONTAINER TO LB
6. REMOVE OLD CONTAINER FROM LB
7. STOP OLD CONTAINER

wait_for: port=8080 delay=5 timeout=15

@michieltcs
DEPLOYING

1. Pull Image
2. Start New Container
3. Wait for Port
4. Smoke Tests / Health Checks
   - uri:
     - url: http://localhost:8080/health
     - status_code: 200
     - timeout: 30
5. Add New Container to LB
6. Remove Old Container from LB
7. Stop Old Container
PULL IMAGE

template: src=haproxy.cfg.j2
dest=/etc/haproxy/haproxy.cfg

service: name=haproxy state=reloaded

ADD NEW CONTAINER TO LB

REMOVE OLD CONTAINER FROM LB

STOP OLD CONTAINER
DEPLOYING

PULL IMAGE

START NEW CONTAINER

ADD NEW CONTAINER TO LB

REMOVE OLD CONTAINER FROM LB

STOP OLD CONTAINER

template: src=haproxy.cfg.j2
dest=/etc/haproxy/haproxy.cfg

service: name=haproxy state=reloaded
DEPLOYING

PULL IMAGE

START NEW CONTAINER

WAIT FOR PORT

SMOKE TESTS / HEALTH CHECKS

ADD NEW CONTAINER

REMOVE OLD CONTAINER

STOP OLD CONTAINER

```bash
docker stop
docker rm
```
# Build Pipeline

Average stage times:
(Average full run time: ~3min 31s)

<table>
<thead>
<tr>
<th>#9</th>
<th>Jun 10</th>
<th>10:41</th>
<th>No Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>#8</td>
<td>Jun 10</td>
<td>10:32</td>
<td>No Changes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run tests</td>
<td>39s</td>
</tr>
<tr>
<td>Build docker image</td>
<td>41s</td>
</tr>
<tr>
<td>Deploy acceptance</td>
<td>53s</td>
</tr>
<tr>
<td>Deploy production</td>
<td>1min 17s</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Run tests</td>
<td>46s</td>
</tr>
<tr>
<td>Build docker image</td>
<td>51s</td>
</tr>
<tr>
<td>Deploy acceptance</td>
<td>1min 12s</td>
</tr>
<tr>
<td>Deploy production</td>
<td>2min 9s</td>
</tr>
</tbody>
</table>

master
master
master
master
master
master
master
master
FEEDBACK!
RESULTS
Build per service < 10 min.
1. Build per service < 10 min.
2. 50+ deploys per day

@michielts
1. Build per service < 10 min.
2. 50+ deploys per day
3. Reduced number of issues
1. Build per service < 10 min.
2. 50+ deploys per day
3. Reduced number of issues
4. Improved page load times
Improved metrics & audience statistics
Improved metrics & audience statistics

Learning new technology
Improved metrics & audience statistics

Learning new technology

Increased confidence, velocity & fun
Team acceptance
Team acceptance

New technology
Team acceptance
New technology
Docker stability
<table>
<thead>
<tr>
<th>Team acceptance</th>
</tr>
</thead>
<tbody>
<tr>
<td>New technology</td>
</tr>
<tr>
<td>Docker stability</td>
</tr>
<tr>
<td>Pipeline stability</td>
</tr>
</tbody>
</table>
Feature toggle cap
Feature toggle cap

Business alignment
Feature toggle cap

Business alignment

Focus on replacing legacy
Reduce cycle times. React faster to the market and test product ideas. Make things!

We make Agile, DevOps and Continuous Delivery accessible!

Hands-on, results-oriented approach to get you where you need to be.

Modern infrastructure and pipelines in minutes.
THANK YOU!

@michieltcs / michiel@make.io

www.michielrook.nl