7 things which you should care about before release your code to production

Mateusz Dymiński
Nokia

github.com/mateuszdyminski/7things-java (github.com/mateuszdyminski/7things-java)

Whoami

Mateusz Dymiński:

- Software Developer at Nokia
- 7+ exp with Java
- 3+ exp with Go
- One of the organizer GoWroc - Golang Wroclaw Meetup (https://www.meetup.com/GoWroc)
- Github: github.com/mateuszdyminski (http://github.com/mateuszdyminski)
- Twitter: @m_dyminski (http://twitter.com/m_dyminski)
- LinkedIn: linkedin.com/in/mdyminski (http://linkedin.com/in/mdyminski)

Agenda

- Versioning
- Profiling
- Health checks
- Logs
- Performance
- Monitoring and alarming
- Release process

1. Versioning

Version Builds

Version Builds

Add information about the build at build time.

- Version
- Last commit
- Build time
- Env vars

Version Builds - how to

Use maven plugin to add information about build

https://github.com/ktoso/maven-git-commit-id-plugin (https://github.com/ktoso/maven-git-commit-id-plugin)

Version Builds - usage

Get information about the build and last commit from git.

```
<plugins>
    <plugin>
        <groupId>pl.project13.maven</groupId>
        <artifactId>git-commit-id-plugin</artifactId>
        <version>2.2.2</version>
        <executions>
            <execution>
                <id>get-the-git-infos</id>
                <goals>
                    <goal>revision</goal>
                </goals>
            </execution>
            <execution>
                <id>validate-the-git-infos</id>
                <goals>
                    <goal>validateRevision</goal>
                </goals>
                <phase>package</phase>
            </execution>
        </executions>
    </plugin>
</plugins>
```

Version Builds

Plugin adds `git.properties` file to jar

```
#Generated by Git-Commit-Id-Plugin
#Fri Jun 30 22:20:44 CEST 2017
git.build.user.email=
git.build.host=md
git.dirty=true
git.remote.origin.url=https\://github.com/mateuszdyminski/7things-java.git
git.closest.tag.name=
git.commit.id.describe-short=ce7b0e9-dirty
git.commit.user.email=dyminski@gmail.com
git.commit.time=29.06.2017 @ 15\:40\:30 CEST
git.commit.message.full=Initial commit
git.build.version=1.0-SNAPSHOT
git.commit.message.short=Initial commit
git.commit.id.abbrev=ce7b0e9
git.branch=master
git.build.user.name=
git.closest.tag.commit.count=
git.commit.id.describe=ce7b0e9-dirty
git.commit.id=ce7b0e9306ad46c011b97d5c51359fc1911a5231
git.tags=
git.build.time=30.06.2017 @ 22\:20\:44 CEST
git.commit.user.name=Mateusz Dyminski
```

Version Builds

We should load it and make available over Health endpoint

```
    private GitRepositoryState getGitRepositoryState() {
        try {
            Properties properties = new Properties();
            properties.load(getClass().getClassLoader().getResourceAsStream("git.properties"));
            return new GitRepositoryState(properties);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
```

```
@Path("health")
public class HealthResource {

    @Inject
    private GitRepositoryState gitRepositoryState;

    @GET
    @Produces("application/json")
    public Response getHealth() {
        return Response.ok().entity(new HealthStatus(gitRepositoryState)).build();
    }
}
```

Version Builds - demo

http://localhost:8090/health (http://localhost:8090/health)

Version artifacts

Version artifacts

Always store build results.

- Nexus, ftp or something
- Github, Bitbucket, Codeplex
- Dockerhub, private Docker registry

Version artifacts - github - how to

```
mvn deploy:deploy-file -Durl=file://~\m2-repo \
                       -DrepositoryId=some.id \
                       -Dfile=your-artifact-1.0.jar \
                       [-DpomFile=your-pom.xml] \
                       [-DgroupId=org.some.group] \
                       [-DartifactId=your-artifact] \
                       [-Dversion=1.0] \
                       [-Dpackaging=jar] \
                       [-Dclassifier=test] \
                       [-DgeneratePom=true] \
                       [-DgeneratePom.description="My Project Description"] \
                       [-DrepositoryLayout=legacy] \
                       [-DuniqueVersion=false]
```

Version artifacts - docker - how to

```
<plugin>
  <groupId>com.spotify</groupId>
  <artifactId>docker-maven-plugin</artifactId>
  <configuration>
    <imageName>my-image</imageName>
    ...
  </configuration>
  <executions>
    <execution>
      <id>build-image</id>
      <phase>package</phase>
      <goals>
        <goal>build</goal>
      </goals>
    </execution>
    <execution>
      <id>tag-image</id>
      <phase>package</phase>
      <configuration>
        <image>my-image</image>
        <newName>registry.example.com/my-image</newName>
      </configuration>
    </execution>
  </executions>
</plugin>
```

Version artifacts - docker - how to

```
mvn clean package docker:build -DpushImage
```

Version artifacts - github - how to

https://github.com/aktau/github-release (https://github.com/aktau/github-release)

```
# upload a file, for example the OSX binary
$ github-release upload \
    --user mateuszdyminski \
    --repo 7things \
    --tag v0.1.0 \
    --name "myapp" \
    --file bin/darwin/myapp
```

Version API

Version API

Two common approach:

- http://company.com/api/v2/users

```
GET v2.0/users HTTP/1.1
Accept: application/json
```

or:

- http://company.com/api/users

```
GET /users HTTP/1.1
Accept: application/vnd.usersapp-v2+json
```

or:

- http://company.com/api/users

```
GET /users HTTP/1.1
x-app-version: 2.0
```

Version API - path-based

- http://company.com/api/v2/users

```
GET v2/users HTTP/1.1 Accept: application/json
```

Code:

```
@Path("/api")
public class UsersResource {

    @GET
    @Path("/v{version}/users")
    @Produces("application/json")
    public Response getUsers(@PathParam("version") String version) {
        if (version.equalsIgnoreCase("1")) {
            return Response.ok()
                .entity(Arrays.asList(new User("Jan", "Kowalski")))
                .build();
        } else {
            return Response.ok()
                .entity(Arrays.asList(new User("Jan", "Kowalski", "987654321")))
                .build();
        }
    }
}
```

Version API - how to

Test:

http://localhost:8180/api/v1/users (http://localhost:8180/api/v1/users)

http://localhost:8180/api/v2/users (http://localhost:8180/api/v2/users)

Version API - headers - how to

- http://company.com/api/users

```
GET /users HTTP/1.1 Accept: application/vnd.usersapp-v2+json
```

Code:

```
@Resource
@Path("users")
public class UsersResource {

    @GET
    @Produces("application/vnd.usersapp-v1+json")
    public Response getUsersV1() {
        User user = new User("Jan", "Kowalski");
        return Response.ok().entity(Arrays.asList(user)).build();
    }

    @GET
    @Produces("application/vnd.usersapp-v2+json")
    public Response getUsersV2() {
        User user = new User("Jan", "Kowalski", "987654321");
        return Response.ok().entity(Arrays.asList(user)).build();
    }
}
```

Version API - headers - how to

Test:

```
curl http://localhost:8180/users

curl -H "Accept: application/vnd.usersapp-v1+json" http://localhost:8180/users

curl -H "Accept: application/vnd.usersapp-v2+json" http://localhost:8180/users
```

Version Static files

Version Static files

Reduce the risk that client gets cached-old files.

- /statics/$REVISION/app.js
- /statics/app.js?v=$REVISION
- /statics/app.$REVISION.js

Version DB schema

Version DB schema

- Automatic - one command to run all migrations
- Reversible - rollbacks

Options:

- Flyway
- Liquibase
- Custom solution

Version DB - Liquibase example

changelog.sql

```
--liquibase formatted sql

--changeset mdyminski:1 dbms:mssql
--comment create users table
create table users (
    id int primary key,
    name varchar(255)
);
--rollback drop table users;

--changeset mdyminski:2 dbms:mssql
--comment alter table users - add new field
alter table users add phone nvarchar (255) NULL;
--rollback ALTER TABLE users DROP COLUMN phone;
```

it also might be xml, yaml, json

Version DB - how to run

```
# /bin/bash

liquibase --changeLogFile=changelog.sql
     --driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
     --classpath=jdbc/sqljdbc4-4.0.jar
     --url="jdbc:sqlserver://somedb.database.windows.net:1433;database=some-db"
     --username=some-user
     --password=some-password
     update
```

it also might be migrate, validate, diff, sql output

2. Profiling

Profiling

Add a way to profile your application in any time.

Sometimes you have to switch something on to be able to profile your application during the normal production shift.

Profiling

Options:

- Your Kit
- Java Mission Control
- JProfiler
- VisualVM
- NetBeans profiler
- Jprobe

Profiling

To enable remote access to our Java app add flags:

```
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=8011
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.authenticate=false
```

Don't forget about the firewall/security policy

Profiling

Don't forget about the

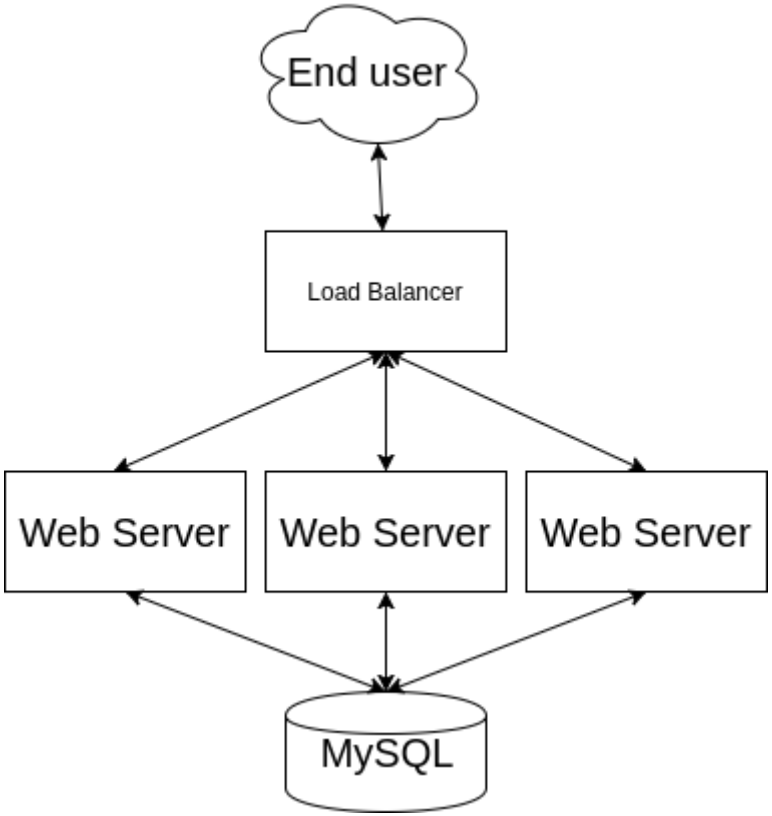
```
-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/some/place/dumps
```

3. Health checks

Health checks

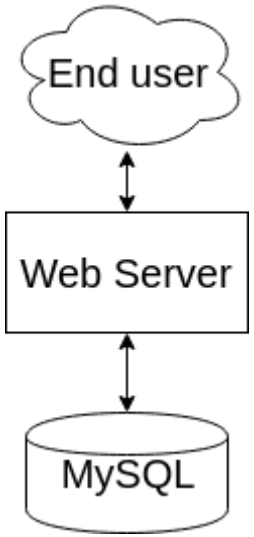
Do I need health check ?

Health checks



Health checks

Do I need health checks in following architecture?



Healthz

- It leverage health endpoint pattern.
- Checks connection to the DB.
- Prints much more information about the service health like:

```
- Build info
- Uptime
- Hostname
- Db connection status
```

Inspiration:

GitHub - app-healthz (https://github.com/kelseyhightower/app-healthz)

Kelsey Hightower - healthz (https://vimeo.com/173610242)

Health checks - how to

We could reuse pattern from Versioning Builds:

```
    private GitRepositoryState getGitRepositoryState() {
        try {
            Properties properties = new Properties();
            properties.load(getClass().getClassLoader().getResourceAsStream("git.properties"));
            return new GitRepositoryState(properties);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
```

Health checks - how to

```
@Path("api/health")
public class ExtendedHealthResource {

    @Inject
    private GitRepositoryState gitRepositoryState;

    @Inject
    private MongoDatabase mongoDatabase;

    @Inject
    private AppStatus appStatus;

    @GET
    @Produces("application/json")
    public Response getHealth() {
        Document serverStatus = mongoDatabase.runCommand(new Document("serverStatus", 1));
        return Response.ok()
            .entity(new ExtendedHealthStatus(gitRepositoryState, serverStatus, appStatus))
            .build();
    }
}
```

Health checks - how to

```
public class AppStatus {

    private String hostname;
    private LocalDateTime StartedAt;

    public AppStatus(String hostname, LocalDateTime startedAt) {
        this.hostname = hostname;
        StartedAt = startedAt;
    }

    public String getHostname() {
        return hostname;
    }

    @JsonProperty
    public String getStartedAt() {
        return StartedAt.toString();
    }

    @JsonProperty
    public String uptime() {
        return Duration.between(StartedAt , LocalDateTime.now()).toString();
    }
}
```

Health checks - how to

```
    private AppStatus getAppStatus() {
        String hostname = "Unknown";

        try
        {
            InetAddress addr;
            addr = InetAddress.getLocalHost();
            hostname = addr.getHostName();
        }
        catch (UnknownHostException ex)
        {
            throw new RuntimeException(ex);
        }

        return new AppStatus(hostname, LocalDateTime.now());
    }
```

Health checks - Demo

Open

http://localhost:8090/api/health (http://localhost:8090/api/health)

4. Logs

Logs

- Use structured logger
- Log context of invocation
- Use log aggregators
- Log with error level should force user (administrator, or direct user) intervention!

Logs - simple example

```
@Path("calculator")
public class Calculator {

    private static Logger logger = LogManager.getLogger(Calculator.class);

    @GET
    @Path("add")
    @Produces("text/plain")
    public String add(@QueryParam("val1") Integer val1,
                      @QueryParam("val2") Integer val2) {
        if (val1 != null) {
            logger.info("Got val1: {}", val1);
        }

        if (val2 != null) {
            logger.info("Got val2: {}", val2);
        }

        int result = val1 + val2;
        logger.info("Result: {}", result);

        return format("Result of %d + %d = %d", val1, val2, result);
    }
}
```

Logs - demo

Run

```
curl 'http://localhost:8070/calculator/add?val1=3&val2=7'
```

Got result

```
Result of 3 + 7 = 10
```

Logs

```
[INFO] 22:31:54.649 LF.filter - HTTP REQUEST: GET /calculator/add Headers: {host=[localhost:8070], user-
[INFO] 22:31:54.653 C.add - Got val1: 3
[INFO] 22:31:54.654 C.add - Got val2: 7
[INFO] 22:31:54.655 C.add - Result: 10
[INFO] 22:31:54.656 LF.filter - HTTP RESPONSE: GET /calculator/add Status: 200
```

Logs - demo v2

Run

```
hey -n 100 -c 10 -H "Content-type: text/plain" "http://localhost:8070/calculator/add?val1=3&val2=7"
```

Logs

```
[INFO] 22:36:24.458 LF.filter - HTTP RESPONSE: GET /calculator/add  Status: 200
[INFO] 22:36:24.458 C.add - Got val2: 7
[INFO] 22:36:24.459 C.add - Result: 10
[INFO] 22:36:24.459 LF.filter - HTTP RESPONSE: GET /calculator/add  Status: 200
[INFO] 22:36:24.461 LF.filter - HTTP REQUEST: GET /calculator/add  Headers: {host=[localhost:8070], user
[INFO] 22:36:24.461 LF.filter - HTTP REQUEST: GET /calculator/add  Headers: {host=[localhost:8070], user
[INFO] 22:36:24.462 C.add - Got val1: 3
[INFO] 22:36:24.462 C.add - Got val2: 7
[INFO] 22:36:24.462 C.add - Got val1: 3
[INFO] 22:36:24.462 C.add - Result: 10
[INFO] 22:36:24.462 C.add - Got val2: 7
[INFO] 22:36:24.462 LF.filter - HTTP RESPONSE: GET /calculator/add  Status: 200
[INFO] 22:36:24.463 C.add - Result: 10
[INFO] 22:36:24.463 LF.filter - HTTP RESPONSE: GET /calculator/add  Status: 200
```

Logs - structured logger + context

```
@PreMatching
public class LoggingFilterWithCtx implements ContainerRequestFilter, ContainerResponseFilter {
    private static Logger logger = LogManager.getLogger(LoggingFilter.class);

    @Inject
    private Provider<Request> request;

    @Override
    public void filter(ContainerRequestContext requestContext) throws IOException {
        ThreadContext.put("requestId", UUID.randomUUID().toString());
        ThreadContext.put("requestIp", request.get().getRemoteAddr());
        logger.info("HTTP REQUEST: {} /{}  Headers: {}",
                requestContext.getMethod(), requestContext.getUriInfo().getPath(),
                requestContext.getHeaders());
    }

    @Override
    public void filter(ContainerRequestContext requestContext, ContainerResponseContext responseContext)
        logger.info("HTTP RESPONSE: {} /{}  Status: {}",
                requestContext.getMethod(), requestContext.getUriInfo().getPath(),
                responseContext.getStatus());
        ThreadContext.remove("requestId");
        ThreadContext.remove("requestIp");
    }
}
```

Logs - demo v3

Run

```
hey -n 100 -c 10 -H "Content-type: text/plain" "http://localhost:8080/calculator/add?val1=3&val2=7"
```

Logs

```
[I] 22:39:16.19 98c3d2f1-815e216e0c63 [grizzly-11] 127.0.0.1 - HTTP RESPONSE: GET /calculator/add  Statu
[I] 22:39:16.19 aab9e772-f9abdf62a129 [grizzly-9] 127.0.0.1 - Got val1: 3
[I] 22:39:16.19 aab9e772-f9abdf62a129 [grizzly-9] 127.0.0.1 - Got val2: 7
[I] 22:39:16.20 aab9e772-f9abdf62a129 [grizzly-9] 127.0.0.1 - Result: 10
[I] 22:39:16.19 a589072c-fa04f4fa2a3b [grizzly-12] 127.0.0.1 - Got val2: 7
[I] 22:39:16.20 a589072c-fa04f4fa2a3b [grizzly-12] 127.0.0.1 - Result: 10
[I] 22:39:16.20 aab9e772-f9abdf62a129 [grizzly-9] 127.0.0.1 - HTTP RESPONSE: GET /calculator/add  Status
[I] 22:39:16.20 9568307d-9ace8480f489 [grizzly-14] 127.0.0.1 - HTTP REQUEST: GET /calculator/add  Header
[I] 22:39:16.21 9568307d-9ace8480f489 [grizzly-14] 127.0.0.1 - Got val1: 3
[I] 22:39:16.22 9568307d-9ace8480f489 [grizzly-14] 127.0.0.1 - Got val2: 7
[I] 22:39:16.22 9568307d-9ace8480f489 [grizzly-14] 127.0.0.1 - Result: 10
[I] 22:39:16.22 9568307d-9ace8480f489 [grizzly-14] 127.0.0.1 - HTTP RESPONSE: GET /calculator/add  Statu
[I] 22:39:16.21 29a0e614-d536ec677462 [grizzly-13] 127.0.0.1 - HTTP REQUEST: GET /calculator/add  Header
[I] 22:39:16.22 a589072c-fa04f4fa2a3b [grizzly-12] 127.0.0.1 - HTTP RESPONSE: GET /calculator/add  Statu
[I] 22:39:16.23 29a0e614-d536ec677462 [grizzly-13] 127.0.0.1 - Got val1: 3
[I] 22:39:16.24 29a0e614-d536ec677462 [grizzly-13] 127.0.0.1 - Got val2: 7
[I] 22:39:16.24 29a0e614-d536ec677462 [grizzly-13] 127.0.0.1 - Result: 10
```

Log things in the frontend/mobile app

- Add extra endpoint in the backend for logs from frontend - tricky and risky
- Use Sentry or other similar tool
- Sometimes backend works as expected but end user is suffering

Use log aggregators

In case where you would like to have more that 1 node or you don't know the flags for grep :)

Improve security - hacker can clean up local files but can't remove logs which are already sent to the log aggregators.

Options:

- elasticsearch
- splunk - if you're rich enough
- greylog
- loggly and many more

Use log aggregators - ELK -how to

Configure log4j to send logs to LogStash - old way

```
log4j.rootLogger=debug,tcp

log4j.appender.tcp=org.apache.log4j.net.SocketAppender
log4j.appender.tcp.Port=5000
log4j.appender.tcp.RemoteHost=localhost
log4j.appender.tcp.ReconnectionDelay=10000
log4j.appender.tcp.Application=playground
```

Use log aggregators - ELK -how to

Use http://www.fluentd.org/ (http://www.fluentd.org/)

```
<source>
 @type tail
 format apache
 tag apache.access
 path /var/log/httpd/access_log
</source>

<match var.log.containers.log**>
    type elasticsearch
    log_level info
    host elasticsearch-logging
    port 9200
    logstash_format true
    num_threads 8
 </match>
```

Use log aggregators - ELK -how to

Use FileBeats https://www.elastic.co/products/beats/filebeat (https://www.elastic.co/products/beats/filebeat)

```
# filebeat.yml
filebeat:
  prospectors:
    -
      paths:
        - /var/log/your-app/app.*.log
      input_type: log
output:
  logstash:
    hosts: ["your-logstash-host:5000"]
```

5. Performance

Performance

- Run performance tests at least once and save the results.
- Run stability tests at least once.
- Don't forget to run them on environment similar to production.
- This isn't as much of time consuming as many think.

Options:

- http://gatling.io/ (http://gatling.io/)
- http://jmeter.apache.org/ (http://jmeter.apache.org/)
- https://github.com/tsenart/vegeta (https://github.com/tsenart/vegeta)
- https://github.com/rakyll/boom (https://github.com/rakyll/boom) / https://github.com/rakyll/hey (https://github.com/rakyll/hey)

Performance - report

```
hey -c 10 -n 1000 http://localhost:8090/api/health
```

Result:

```
Summary:
  Total:    4.1074 secs
  Slowest:    0.2727 secs
  Fastest:    0.0353 secs
  Average:    0.0389 secs
  Requests/sec:    243.4658

Status code distribution:
  [200]    1000 responses

Response time histogram:
  0.035 [1]    |
  0.059 [969]    |■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  0.083 [24]    |■
  0.107 [5]    |
  0.130 [0]    |
  0.154 [0]    |
  0.178 [0]    |
  0.202 [0]    |
  0.225 [0]    |
```

6. Monitoring and alarming

Monitoring and alarming

- Monitoring should never require a human to interpret any part of the alerting domain
- Three valid kinds of monitoring output

```
- Alerts: human needs to take action immediately
- Tickets: human needs to take action eventually
- Logging: no action needed
```

- Eliminating toil: Carla Geisser: “If a human operator needs to touch your system during normal operations, you have a bug. The definition of normal changes as your systems grow.”

```
- If you have to ssh to your server to do some work - something is wrong
- Automate backups
- Automate logs rotating - send old ones to S3/equivalent
```

Monitoring

- Measure everything
- Set alerts based on the metrics
- Analytics can use your metrics!

Monitoring - key things to measure

- Java GC metrics
- Http code rates
- Http response time percentiles - 50, 95, 99
- Error logs
- Uptime
- Business metrics
- IPC(instructions per cycle) over CPU - CPU utilization is wrong (http://www.brendangregg.com/blog/2017-05-09/cpu-utilization-is-wrong.html)

Monitoring - Librato - how to

```
MetricRegistry registry = environment.metrics();
Librato.reporter(registry, "<Librato Email>", "<Librato API Token>")
    .setSource("<Source Identifier (usually hostname)>")
    .start(10, TimeUnit.SECONDS);
```

```
Librato.metric(registry, "logins").tag("uid", uid).meter().mark()
Librato.metric(registry, "kafka-read-latencies").tag("broker", broker).histogram().update(latency)
Librato.metric(registry, "temperature").source("celcius").tag("type", "celcius").gauge(() -> 42))
Librato.metric(registry, "jobs-processed").source("ebs").meter().mark()
Librato.metric(registry, "just-these-tags").tag('"foo", "bar").doNotInheritTags().timer.update(time)
```

7. Release process

Release process - release notes

Create release notes

- JIRA - How to get release notes from JIRA (https://confluence.atlassian.com/adminjiraserver071/creating-release-notes-802592502.html)
- GIT - git log v1.0..v1.2

Create release plan

- When release the application (time, day)
- What if something goes wrong? Revert?

Release process - Add release automation

- Ansible/Puppet/Chef/Salt*
- Docker/Kubernetes/Swarm
- Sometimes bash is good enough :)
- Use process daemons - Does my app be up & running when node will be rebooted?

```
Use: systemd/supervisor/runit
```

Release process - no downtime(or minimize it)

- Graceful shutdown
- Canary deployments
- Rollout deployments - kubernetes

Thank you

Mateusz Dymiński
Nokia

github.com/mateuszdyminski/7things-java (github.com/mateuszdyminski/7things-java)
@m_dyminski (http://twitter.com/m_dyminski)