Pragmatic Continuous Delivery

Day 1
Continuous Delivery 101
Scrum vs Continuous Deployment or why Scrum falls short...

May 10, 2011 - Scrum vs Continuous Deployment or why Scrum falls short for web applications ... The basic idea of Scrum is to create a safe and change-free ...
devops engineer
devops engineer salary
devops engineer resume
devops engineer skills
devops engineer jobs
devops engineer interview questions
DevOps

Kanban

Continuous Integration

Continuous Testing

Continuous Delivery

Continuous Deployment
DevOps

Kanban

Continuous Integration

Continuous Testing

Continuous Delivery

Continuous Deployment
DevOps
Kanban

**Continuous Integration**

Continuous Testing
Continuous Delivery
Continuous Deployment
DevOps
Kanban
Continuous Integration

**Continuous Testing**

Continuous Delivery
Continuous Deployment
What?

A methodology for reducing the cost, time and risk of delivering incremental changes to users.

Qualities

1. Software is always in shippable state once code is pushed into the mainline (including infrastructure, configuration, data)

2. Push-button deployment for any desired version
DevOps
Kanban
Continuous Integration
Continuous Testing
Continuous Delivery

Continuous Deployment
Why Continuous Delivery?
High-performing organizations are deploying code 30 times more frequently, with 50 percent fewer failures than their lower-performing counterparts.

Amazon

new code is deployed every 11.6 seconds during a normal business day (3K production deployments per day)
Facebook

each of 5,000 engineers commits to trunk HEAD at least once a day and the code at trunk HEAD is pushed to production once daily
Etsy

50 deploys/day
Google

15K engineers work from the HEAD revision of a single Perforce trunk. 50% of the code will be changed in any given month. 8 minutes after you commit code it’s live in production.
This book integrates into a compelling narrative the best current thinking about how to create great software-intensive products and services. The approach in this book is both challenging and disciplined, and some organizations will be unable to imagine following this path. But those who make the journey will find it impossible to imagine ever going back—and if they happen to be a competitor, they are well positioned to steal both your market and your people. Ignore this book at your own risk.

(c) Mary Poppendieck
Anatomy of Deployment Pipeline
All changes to production go through deployment pipeline
All changes to production go through version control (from mainline!)
Build only once

Mainline → Commit Stage → Automated Acceptance Testing → Exploratory Testing / UAT → Production Deployment
Test on production-like environment
Deploy the same way to every environment
Commit Stage
Continuous Integration compliance checklist

1. All developers push the code at least once a day (to Mainline)

2. All developers run tests locally before pushing the code (and never push the code if tests fail)

3. **Every change** results in a build and tests run

4. Developers never push the code if a build is broken (why the build is broken if #3 is true?)

5. Build is **always** fixed within ten minutes of it going red
I will work in a branch and sync with Mainline every day.

(and push when my feature is ready)
I will use Mainline as a primary tool for identifying regression in my code
CI changes dynamics of a game

- No painful merges (try aggressive refactoring w/o CI)
- Small increments (easier code review, more opportunities for pairing)
- Evergreen Mainline requires engineering rigor (TDD, preflight quality control)
- Feature branching becomes unnecessary (brings back synchronous code reviews)
- Faster feedback from Sheriff on Duty (SoD)
- Faster feedback from downstream quality gates (if any)
One thing that I really like about open-source is that it really allows different people to work together. We don't have to like each other. And sometimes we really don't like each other.

(c) Linus Torvalds

http://www.ted.com/talks/linus_torvalds_the_mind_behind_linux
Dealing with unfinished functionality
Feature Toggles
Feature Toggles

- Release Toggles
- Business Toggles
Use cases

- Decoupling deployment from release
- Enabling feature for subgroup of users
- A/B testing
- Addition to circuit breaking
Feature Flipping for Java

FF4J is a proposition of Feature Toggle. You can enable and disable features through configuration at runtime with dedicated consoles or Web API but also monitor features usage. You can also define any Property and change its value at runtime with the exact same web console.

More information at ff4j.org or reference guide. To access a demo please click here.

Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Group</th>
<th>Permissions</th>
<th>Strategy</th>
<th>Toggle</th>
<th>E</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>mercury-1nk</td>
<td>mercury</td>
<td>ROLE_USER</td>
<td>org.ff4j.strategy.DecisionStrategy {weight=0.5}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mars-2nc</td>
<td>mars</td>
<td>ROLE_USER</td>
<td>--</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>saturn-3nk</td>
<td>sature</td>
<td>ROLE_USER</td>
<td>--</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>Group</td>
<td>Permissions</td>
<td>Strategy</td>
<td>Toggle</td>
<td>E</td>
<td>D</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>-------------</td>
<td>----------</td>
<td>--------</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>earth-desc</td>
<td>earth</td>
<td>ROLE_USER</td>
<td>--</td>
<td>ON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>earth-link</td>
<td>earth</td>
<td>ROLE_USER</td>
<td>--</td>
<td>ON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mercury-link</td>
<td>mercury</td>
<td>ROLE_USER</td>
<td>--</td>
<td>ON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mercury-desc</td>
<td>mercury</td>
<td>ROLE_USER</td>
<td>--</td>
<td>ON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>venus-link</td>
<td>venus</td>
<td>ROLE_USER</td>
<td>--</td>
<td>ON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>venus-desc</td>
<td>venus</td>
<td>ROLE_USER</td>
<td>--</td>
<td>ON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>jupiter-link</td>
<td>jupiter</td>
<td>ROLE_USER</td>
<td>--</td>
<td>ON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>jupiter-desc</td>
<td>jupiter</td>
<td>ROLE_USER</td>
<td>--</td>
<td>ON</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
if (ff4j.exist("new-feature")) {
    // new-feature exists
}

if (ff4j.check("new-feature")) {
    // new-feature is toggled
}
Launch, control, and measure your features

Try it Free  Request a Demo

Watch The Video

TRUSTED AND USED BY

Microsoft  AppDirect  circleci  TEN-X  BEHALF
Anti-pattern: Feature Toggles introduce additional failure mode
Release toggles are a useful technique and lots of teams use them. However they should be your last choice when you're dealing with putting features into production. Your first choice should be to break the feature down so you can safely introduce parts of the feature into the product. The advantages of doing this are the same ones as any strategy based on small, frequent releases. You reduce the risk of things going wrong and you get valuable feedback on how users actually use the feature that will improve the enhancements you make later.

(c) Martin Fowler at Bliki
Breaking changes
Rule: In Continuous Delivery there are no breaking changes
Branch by Abstraction vs. Branch by Source Control
Consumer

Component to be replaced

Consumer
Consumer -> Component to be replaced
Consumer -> Abstraction
Abstraction -> Component to be replaced
Consumer -> Abstraction
Abstraction -> New version of component
Abstraction -> Component to be replaced
Strangler Application

Martin Fowler
29 June 2004

When Cindy and I went to Australia, we spent some time in the rain forests on the Queensland coast. One of the natural wonders of this area are the huge strangler vines. They seed in the upper branches of a fig tree and gradually work their way down the tree until they root in the soil. Over many years they grow into fantastic and beautiful shapes, meanwhile strangling and killing the tree that was their host.

This metaphor struck me as a way of describing a way of doing a rewrite of an important system. Much of my career has involved rewrites of critical systems. You would think such a thing as easy - just make the new one do what the old one did. Yet they are always much more complex than they seem, and overflowing with risk. The big cut-over date looms, the pressure is on. While new features (there are always new features) are liked, old stuff has to remain. Even old bugs often need to be added to the rewritten system.

An alternative route is to gradually create a new system around the edges of the old, letting it grow slowly over several years until the old system is strangled. Doing this sounds hard, but increasingly I think it's one of those things that isn't tried enough. In particular I've noticed a couple of basic strategies that work well. The fundamental strategy is Event Interception, which can be used to gradually move functionality to the strangler and to enable Asset Capture.
### Upcoming changes

<table>
<thead>
<tr>
<th>Estimated date</th>
<th>Description</th>
<th>Announcement</th>
<th>Affected APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>March 5, 2013</td>
<td>Retirement of deprecated @Anywhere API</td>
<td>Original blog post</td>
<td>@Anywhere API</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated deadline announcement</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Retirement of deprecated API v1.0.</td>
<td>Deprecation announcement Release of 1.1</td>
<td>REST API v1.0 Streaming API v1.0</td>
</tr>
</tbody>
</table>

### Recent changes

<table>
<thead>
<tr>
<th>Deployed date</th>
<th>Description</th>
<th>Announcement</th>
<th>Affected APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 5, 2012</td>
<td>Sitestreams certificate change. Blog post</td>
<td></td>
<td>Sitestreams API v1.0 Sitestreams API v1.1</td>
</tr>
<tr>
<td>November 2, 2012</td>
<td>Changes to withheld content fields. Blog post</td>
<td></td>
<td>REST API v1.0 REST API v1.1 Streaming API v1.0 Streaming API v1.1</td>
</tr>
<tr>
<td>October 25, 2012</td>
<td>Entities in streamed DMs. Announcement</td>
<td></td>
<td>Sitestreams API v1.0 Sitestreams API v1.1 Userstreams API v1.0 Userstreams API v1.1</td>
</tr>
</tbody>
</table>
Meaningful commits
4W
Who
When
What
Why
How
$ git log --oneline -5 --author cbeams --before "Fri Mar 26 2009"

e5f4b49 Re-adding AutoConfigurationPostProcessor
2db0f12 fixed two build-breaking issues
147709f Tweaks to dependency files
7f96f57 polishing
2d30f32 implemented facebook integration

$ git log --oneline -5 --author pwebb --before "Sat Aug 30 2014"

5ba3db6 Add automatic configuration with reasonable defaults
84564a0 Improve stability of DateTime tests
e142fd1 Set fixed Guava version from 16.0.* to 16.0.1
ac8326d Polish mockito usage according to [best practices]
2d30f32 Implement facebook authentication [FB-5332]

-> How to Write a Git Commit Message
6 Deadly Sins of A Unit Test
Unstable

OS-specifics

Wildcard dependency versions

Shared state in tests

System Time

Asynchrony

Concurrency

[Doppins] Upgrade dependencies #1

**doppins-bot** wants to merge 34 commits into master from doppins/helpful-initial-upgrade

**doppins-bot** commented 20 seconds ago

Hi, and thank you for trying out Doppins.

This initial pull request upgrades all your dependency ranges to the latest available version. From now on any new dependency releases will result in a pull request to your repository, submitted in real-time.

Make sure that it doesn't break anything, and happy merging! 🤓

The upgraded dependencies are:

- jshint-stylish from \~0.1.3 to \~2.1.0
- load-grunt-tasks from \~0.2.0 to \~3.5.0
- grunt from \~0.4.1 to \~1.0.1
- karma-ng-html2js-preprocessor from \~0.4.1 to \~1.0.0
Testing asynchronous systems is hard. Not only does it require handling threads, timeouts and concurrency issues, but the intent of the test code can be obscured by all these details. Awaitility is a DSL that allows you to express expectations of an asynchronous system in a concise and easy to read manner. For example:

```java
@Test
public void updatesCustomerStatus() throws Exception {
    // Publish an asynchronous event:
    publishEvent(updateCustomerStatusEvent);
    // Awaitility lets you wait until the asynchronous operation completes:
    await().atMost(5, SECONDS).until(customerStatusIsUpdated());
    ...
}
```
public class FlawedList<T> extends ArrayList<T> {
    public boolean putIfAbsent(T object) {
        boolean absent = !super.contains(object);
        if (absent) {
            super.add(object);
        }
        return absent;
    }
}
@Test
public void testPutIfAbsent() {
    FlawedList<String> list = new FlawedList<String>();
    list.putIfAbsent("foo");
    list.putIfAbsent("foo");
    list.putIfAbsent("foo");
    assertThat(list.size(), is(1));
}
FlawedList<String> list = new FlawedList<String>();

@Test(threadPoolSize = 5, invocationCount = 20)
public void testList() {
    list.putIfAbsent("foo");
    assertThat(list.size(), is(1));
}
A Java framework for testing multithreaded code.
public class WeavedFlawedListTest {
    private FlawedList<String> list;

    @ThreadedBefore public void before() {
        list = new FlawedList<String>();
    }

    @ThreadedMain public void mainThread() {
        list.putIfAbsent("foo");
    }

    @ThreadedSecondary public void secondThread() {
        list.putIfAbsent("foo");
    }

    @ThreadedAfter public void after() {
        assertEquals(1, list.size());
    }
}
public class FlawedList<T> extends ArrayList<T> {
    public boolean putIfAbsent(T object) {
        boolean absent = !super.contains(object);
        if (absent) {
            super.add(object);
        }
        return absent;
    }
}
Unreliable

No tests - bad

Bad tests - even worse
Real world mutation testing

PIT is a state of the art mutation testing system, providing gold standard test coverage for Java and the JVM. It’s fast, scalable and integrates with modern test and build tooling.
Slow

Implicit waiting

Computation-intensive

Stupid

Code coverage boosters

Fuzzing

Sequential

Shared state

Sociable

Relying on concrete classes

Relying on external systems
Being sociable is OK, if:

Dependency on non-trivial execution context (e.g. Spring)

File system

SMTP client

SQL repository

Redis repository
Being sociable is OK, if:

Dependency on non-trivial execution context (e.g. Spring)

- File system
- SMTP client
- SQL repository
- Redis repository
Being sociable is OK, if:

- Dependency on non-trivial execution context (e.g. Spring)
- File system
- SMTP client
- SQL repository
- Redis repository

Jimfs

Jimfs is an in-memory file system for

- build passing
- maven central 1.1

Getting started

The latest release is 1.1.

It is available in Maven Central as com.google.jimfs

<dependency>
  <groupId>com.google.jimfs</groupId>
Being sociable is OK, if:

Dependency on non-trivial execution context (e.g. Spring)

File system

SMTP client

SQL repository

Redis repository
Being sociable is OK, if:

- Dependency on non-trivial execution context (e.g. Spring)
- File system
- SMTP client
- SQL repository
- Redis repository
Being sociable is OK, if:

Dependency on non-trivial execution context (e.g. Spring)

File system

SMTP client

SQL repository

Redis repository
Using default tag: latest
latest: Pulling from library/redis

Digest: sha256:3df6902f054108596c35b7e44a774d518b960e42761ea8075ae264ee5ed5f100
Status: Downloaded newer image for redis:latest

Redis 3.0.7 (000000000/0) 64 bit
Running in standalone mode
Port: 6379   Running in standalone mode
PID: 1   PID: 1

http://redis.io
class RedisBackedCacheTest {
  
  @Rule
  public GenericContainer redis = new GenericContainer("redis:3.0.6")
    .withExposedPorts(6379);

  private Cache cache;

  @Before
  public void setUp() {
    Jedis jedis = new Jedis(redis.getIpAddress(), redis.getMappedPort(6379));
    cache = new RedisBackedCache(jedis, "test");
  }

  @Test
  public void findsAnInsertedValueInCache() {
    String key = "foo";
    cache.put(key, "FOO");
    Optional<String> cacheHit = cache.get(key, String.class);
    assertThat(cacheHit, isPresent());
  }

}
class RedisBackedCacheTest {

    @ClassRule
    public static GenericContainer redis = new GenericContainer("redis:3.0.6")
        .withExposedPorts(6379);

    private Cache cache;

    @Before
    public void setUp() {
        Jedis jedis = new Jedis(redis.getIpAddress(), redis.getMappedPort(6379));
        cache = new RedisBackedCache(jedis, "test");
    }

    @Test
    public void findsAnInsertedValueInCache() {
        String key = UUID.randomUUID().toString();
        cache.put(key, "FOO");
        Optional<String> cacheHit = cache.get(key, String.class);
        assertThat(cacheHit, isPresent());
    }
}

WireMock

WireMock is a flexible library for stubbing and mocking web services. Unlike general purpose mocking, it is an actual HTTP server that your code under test can connect to as it would a real web service.

It supports HTTP response stubbing, request verification, proxy/intercept, record/playback of stubs and used from within a unit test or deployed into a test environment.

Although it's written in Java, there's also a JSON API so you can use it with pretty much any language.

Video - Using WireMock in Practice

Rob Elliot and I recently gave a presentation at Skillsmatter to the London Java Community demonstrating WireMock in practice after 4 years of using it.

Watch the video here.

New new new! Version 2.0 in beta

WireMock 2.0 is now in development. We're using the major version bump to make a few breaking changes (don't worry!).

Changes from 1.x include:
class UberSmartHttpClientTest {

@Rule
test public WireMockRule wireMockRule = new WireMockRule(8089);

@Test
test public void exampleTest() {
    stubFor(get(urlEqualTo("/my/resource"))
        .withHeader("Accept", equalTo("text/xml"))
        .willReturn(aResponse()
            .withStatus(200)
            .withHeader("Content-Type", "text/xml")
            .withBody("<response>Some content</response>")));

    Result result = uberSmartHttpClient.doSomeHttpRequest();
    assertTrue(result.wasSuccessful());
}
}
Fault injection

```java
stubFor(get(urlEqualTo("/delayed")).willReturn(
  aResponse()
    .withStatus(200)
    .withFixedDelay(2000));
```
Welcome, friend

mountebank is the first open source tool to provide cross-platform, multi-protocol test doubles over the wire. Simply point your application under test to mountebank instead of the real dependency, and test like you would with traditional stubs and mocks.

mountebank will cure what ails you, guaranteed.

How it works

mountebank employs a legion of *impersonators* to act as on-demand test doubles. Your test communicates to mountebank over http using the api to set up stubs, record and replay proxies, and verify mock expectations. In the typical use case, each test will start an impersonator during test setup and stop an impersonator during test teardown.

mountebank employs several types of impersonators, each responding to a specific protocol. Typically, your test will tell the impersonator which port to bind to, and the impersonator will open the corresponding socket.
Diagram illustrating a software deployment process involving Jenkins, showing steps from push to production deployment.
# docker-compose.yml

version: '2'

services:
  app:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - ./code
    depends_on:
      - redis
      - mysql

redis:
  image: redis:2.8

mysql:
  image: mysql:5.6

...
Recreating compose_redis_1

Pulling redis (redis:2.8)...
2.8: Pulling from library/redis
fdd5d7827f33: Already exists
3868e1e933d6: Already exists
1d007c18c656: Already exists
ad75a8697e9c: Already exists
8de500daf5d7: Pull complete
788efe3bdabf: Pull complete
8f359895dbf8: Pull complete

Digest: sha256:6abf21819a654f91811d6e6e1ccacaf94247d6f417be748501d88782bba8845
Status: Downloaded newer image for redis:2.8

Attaching to compose_redis_1

Warning: no config file specified, using the default config. In order to specify a config file, use docker-compose -f file.conf up

Redis 2.8.23 (00000000/0) 64 bit

Running in stand alone mode
Port: 6379
PID: 1

http://redis.io
<table>
<thead>
<tr>
<th>CONTAINER ID</th>
<th>IMAGE</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>6987c8ba0fff</td>
<td>redis:2.8</td>
<td>&quot;/entrypoint.sh.sh redis&quot;</td>
</tr>
<tr>
<td>0de43f230e0d</td>
<td>redis:2.8</td>
<td>&quot;/entrypoint.sh.sh redis&quot;</td>
</tr>
<tr>
<td>5a025f47c530</td>
<td>redis:2.8</td>
<td>&quot;/entrypoint.sh.sh redis&quot;</td>
</tr>
</tbody>
</table>
FROM nginx
RUN rm -f /etc/nginx/conf.d/*
RUN apt-get update && apt-get install -my \
    supervisor \
    curl \
    wget \
    php5-curl \
    php5-fpm \
    php5-gd \
    php5-memcached \
    php5-mysql \
    php5-ncrypt \
    php5-sqlite \
    php5-xdebug \
    php-apc
RUN sed -i "s/user = www-data/user = root/" /etc/php5/fpm/pool.d/www.conf
RUN sed -i "s/group = www-data/group = root/" /etc/php5/fpm/pool.d/www.conf
RUN sed -i "/^;clear_env = no/s/^//" /etc/php5/fpm/pool.d/www.conf
RUN sed -i "/^;ping\.path/s/^//" /etc/php5/fpm/pool.d/www.conf
RUN sed -i "/^;pm\.status_path/s/^//" /etc/php5/fpm/pool.d/www.conf
DOCKER BUILDER

Type: docker

The docker Packer builder builds Docker images using Docker. The builder starts a Docker container, runs provisioners within this container, then exports the container for reuse or commits the image.

Packer builds Docker containers without the use of Dockerfiles. By not using Dockerfiles, Packer is able to provision containers with portable scripts or configuration management systems that are not tied to Docker in any way. It also has a simpler mental model: you provision containers much the same way you provision a normal virtualized or dedicated server. For more information, read the section on Dockerfiles.
Sysdig is open source, system-level exploration: capture system state and activity from a running Linux instance, then save, filter and analyze. Sysdig is scriptable in Lua and includes a command line interface and a powerful interactive UI, csysdig, that runs in your terminal. Think of sysdig as strace + tcpdump + htop + iftop + lsof + awesome sauce. With state of the art container visibility on top.
YUM repository
- Nexus
- Artifactory
- yum-s3-plugin
- yum-s3-iam

RPM packager
- fpm
- gradle-ospackage-plugin
[program:app]
command=java -port=3000 -logdir=/var/log/app/ -jar /opt/app/current/app.jar
user=deployer
autostart=true
autorestart=true
startsecs=10
startretries=3
stdout_logfile=/var/log/app/stdout.log
stderr_logfile=/var/log/app/stderr.log
WiFi ResetPlug - A smart plug to monitor your WiFi router/modem and reset power if WiFi fails.

by Mullinel

Be the first to review this item

Price: $59.99

In Stock.

This item does not ship to Riga, Latvia. Please check other sellers who may ship internationally. Learn more

Sold by MultiNet and Fulfilled by Amazon.

- Automatically resets power to both the router and broadband modem if the Internet connection is lost.
- LED indicates if the WiFi is working. A solid blue LED means WiFi Internet is working.
- Works with any WiFi router, any WiFi access point, and any broadband modem.
- Also works with any all-in-one combination device (Router/Modem/AP).
- Use a power strip to monitor separate devices (Router/Modem/WiFi AP) with one ResetPlug.
deploy-playbook.yml

- hosts: all
  serial: 1
  tasks:
    - name: install the app
      yum: name=app-{{version}} state=present
      notify:
        - restart supervisord
  handlers:
    - name: restart supervisord
      service: name=supervisord state=restarted
    - name: start the app
      supervisorctl: name=app state=started
    - name: health check
      health_check:
        url: "{{inventory_hostname}}/health"
        delay_between_tries: 5
        max_retries: 20
        expected_regexp: "alive"
  pre_tasks:
    - name: disable nagios alerts for this host webserver service
      nagios: action=disable_alerts host=\{{\ inventory_hostname \} } services=webserver
    - name: disable the server in haproxy
      haproxy: state=disabled host=\{{\ inventory_hostname \} }
    - name: stop the app
      supervisorctl: name=app state=stopped


$ ansible-playbook deploy-playbook.yml -i /hosts/acceptance --extra-vars "version=1.1.2"

$ ansible-playbook deploy-playbook.yml -i /hosts/exploratory --extra-vars "version=1.1.2"

$ ansible-playbook deploy-playbook.yml -i /hosts/production --extra-vars "version=1.1.2"
- bot: building **1.1.2** from commit [add healtchecks] by @eduardsi
- bot: **1.2.2** passed commit stage
- bot: **1.2.2** passed acceptance tests
- bot: **1.1.2** ready to be promoted to exploratory testing

- $ promote 1.1.2

- bot: **1.1.2** is available at [http://exploratory.app.io/1.1.2](http://exploratory.app.io/1.1.2)

- $ promote 1.1.2 --single 50% 100%
- $ features list

- bot: /facebook-registration (disabled)

  /twitter-registration (enabled, 100%)

- $ features enable facebook-registration 10%

- $ features list

- bot: /facebook-registration (enabled 10%)

  /twitter-registration (enabled, 100%)
supervisor.conf

[program:app]
command=java -port=3000 -log.dir=/var/log/app/ -jar /opt/app/current/app.jar
user=deployer
autostart=true
autorestart=true
startsecs=10
startretries=3
stdout_logfile=/var/log/app/stdout.log
stderr_logfile=/var/log/app/stderr.log
Arg4j

@Option(name="-port", usage="HTTP port the application will run on")
public Integer port;
...

@Option(name="-log.dir", usage="A directory where logs will be written to")
public File logDir;
...

$ java -jar app.jar -wrong

"-wrong" is not a valid option
Application [options]
  -port VAL : HTTP port the application will run on
  -logDir FILE : A directory where logs will be written to
deploy-playbook.yml

- hosts: all
  serial: 1
  tasks:
  - name: install the app
    yum: name=app-{{version}} state=present
    notify:
      - restart supervisord
  handlers:
  - name: restart supervisord
    service: name=supervisord state=restarted
  - name: start the app
    supervisorctl: name=app state=started
  - name: health check
    health_check:
      url: "{{inventory_hostname}}/health"
      delay_between_tries: 5
      max_retries: 20
      expected_regexp: "alive"
  pre_tasks:
  - name: disable nagios alerts for this host webserver service
    nagios: action=disable_alerts host={{ inventory_hostname }} services=webserver
  - name: disable the server in haproxy
    haproxy: state=disabled host={{ inventory_hostname }}
  - name: stop the app
    supervisorctl: name=app state=stopped


```json
{
    "counter.status.200.root": 20,
    "counter.status.200.metrics": 3,
    "counter.status.200.star-star": 5,
    "counter.status.401.root": 4,
    "gauge.response.star-star": 6,
    "gauge.response.root": 2,
    "gauge.response.metrics": 3,
    "classes": 5808,
    "classes.loaded": 5808,
    "classes.unloaded": 0,
    "heap": 3728384,
    "heap.committed": 986624,
    "heap.init": 262144,
    "heap.used": 52765,
    "mem": 986624,
    "mem.free": 933858,
    "processors": 8,
    "threads": 15,
    "threads.daemon": 11,
    "threads.peak": 15,
}
```
Part V. Spring Boot Actuator: Production-ready features

Spring Boot includes a number of additional features to help you monitor and manage your application when it's pushed to production. You can choose to manage and monitor your application using HTTP endpoints, with JMX or even by remote shell (SSH or Telnet). Auditing, health and metrics gathering can be automatically applied to your application.

Actuator HTTP endpoints are only available with a Spring MVC-based application. In particular, it will not work with Jersey unless you enable Spring MVC as well.

44. Enabling production-ready features

The [spring-boot-actuator](https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-actuator) module provides all of Spring Boot's production-ready features. The simplest way to enable the features is to add a dependency to the [spring-boot-starter-actuator](https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-actuator) 'Starter POM'.

**Definition of Actuator**

An actuator is a manufacturing term, referring to a mechanical device for moving or controlling something. Actuators can generate a large amount of motion from a small change.

To add the actuator to a Maven based project, add the following 'starter' dependency:

```xml
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
  </dependency>
</dependencies>
```

For Gradle, use the declaration:

```groovy
dependencies {
    compile("org.springframework.boot:spring-boot-starter-actuator")
}
```
Metrics is a Java library which gives you unparalleled insight into what your code does in production.

Metrics provides a powerful toolkit of ways to measure the behavior of critical components in your production environment.

With modules for common libraries like Jetty, Logback, Log4j, Apache HttpClient, Ehcache, JDBI, Jersey and reporting backends like Ganglia and Graphite, Metrics provides you with full-stack visibility.

Getting Started »  User Manual »  About Metrics »
- bot: building **1.1.2** from commit [add healtchecks](https://github.com/your-repo) by @eduardsi

- bot: **1.2.2** passed *commit stage*

- bot: **1.2.2** passed *acceptance tests*

- bot: **1.1.2** ready to be promoted to *exploratory testing*

- `$ promote 1.1.2`

- bot: **1.1.2** is available at [http://exploratory.app.io/1.1.2](http://exploratory.app.io/1.1.2)

- `$ promote 1.1.2 --single 50% 100%`
MAKE LOGGING GREAT AGAIN

All your logs in one place

Centralize and aggregate all your log files for 100% visibility. Use our powerful query language to search through terabytes of log data to discover and analyze important information.

Forward & Archive
Forward subsets of data to other systems in real-time. Archive data you access less frequently to cheaper long-term storage systems.

Monitor & Alert
Trigger actions or get notified when something needs attention, such as failed login attempts, exceptions or performance degradation.

Learn more
Alerts configuration for stream »Exceptions on all platforms«

You can define thresholds on any message field or message count of a stream and be alerted based on this definition.

Learn more about alerts in the documentation.

Add new alert condition

Message count condition

Configure new alert condition

Trigger alert when there are more or less than 0 messages in the last 0 minutes and then wait at least 0 minutes until triggering a new alert. (grace period)

When sending an alert, include the last 0 messages of the stream evaluated for this alert condition.

Add alert condition

Configured alert conditions

Field value condition

Alert is triggered when the field `millis` has a higher mean value than 250 in the last 3 minutes. Grace period: 0 minutes. Not including any messages in alert notification.

Edit condition Delete condition

Callbacks

The following callbacks will be performed when this stream triggers an alert:

Select Callback Type Add callback Find more callbacks

No configured alarm callbacks.
New message in Graylog stream HTTP 500's:

GET /posts/45326 [500] 109ms

Details:

- **Source**: controller
- **resource**: PostsController
- **http_method**: GET
- **action**: http_response_code
- **show**: 500

@channel Alert for Graylog stream HTTP 500's:

Stream had 19 messages in the last 1 minute with trigger condition more than 5 messages. (Current grace time: 0 minutes)

Details:

- **Stream ID**: 55ae6505bee81be853aef467
- **Stream Title**: HTTP 500's
- **Stream Description**: All HTTP 500's
Gor is an open-source tool for capturing and replaying live HTTP traffic into a test environment in order to continuously test your system with real data. It can be used to increase confidence in code deployments, configuration changes and infrastructure changes.

http://leonsbox.com/blog/2013/06/04/improving-testing-by-using-real-traffic-from-production/

<table>
<thead>
<tr>
<th>Branch: master</th>
<th>Latest commit 334db15 2 hours ago</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>byteutils</strong></td>
<td>Improve byteutils</td>
</tr>
<tr>
<td><strong>examples/middleware</strong></td>
<td>Add example of java echo middleware</td>
</tr>
<tr>
<td><strong>proto</strong></td>
<td>Fix tests and formatting</td>
</tr>
<tr>
<td><strong>raw_socket_listener</strong></td>
<td>Merge pull request #253 from buger/listener-tests</td>
</tr>
<tr>
<td><strong>.dockerignore</strong></td>
<td>add .dockerignore</td>
</tr>
</tbody>
</table>
curl -X POST -H "Content-Type: application/json" -d '{"event":"data"}'
localhost:9880/app.request
If there is one thing I miss about SVN having switched to git (and trust me, it’s the only thing), it is the ability to checkout only a sub-tree of a repository. As of version 1.7, you can check out just a sub-tree in git as well! Now not only does git support checking out sub-directories, it does it better than subversion!
Sensitive data?
transparently encrypt files within a git repository

<table>
<thead>
<tr>
<th>elasticdog</th>
<th>Ignore build artifacts within the pacman directory</th>
<th>Latest commit 991a200 on Dec 31, 2015</th>
</tr>
</thead>
<tbody>
<tr>
<td>contrib</td>
<td>Ignore build artifacts within the pacman directory</td>
<td>3 months ago</td>
</tr>
<tr>
<td>man</td>
<td>Add -force option Zsh completion and man page</td>
<td>2 years ago</td>
</tr>
<tr>
<td>.gitattributes</td>
<td>Add encrypted version of a sensitive file</td>
<td>2 years ago</td>
</tr>
<tr>
<td>INSTALL.md</td>
<td>Document installation options in INSTALL.md</td>
<td>3 months ago</td>
</tr>
<tr>
<td>LICENSE</td>
<td>Bump version up to 0.9.7</td>
<td>a year ago</td>
</tr>
<tr>
<td>README.md</td>
<td>Document installation options in INSTALL.md</td>
<td>3 months ago</td>
</tr>
<tr>
<td>sensitive_file</td>
<td>Create helper scripts to simplify filter commands</td>
<td>2 years ago</td>
</tr>
<tr>
<td>transcrypt</td>
<td>Add ability to dump the raw commit objects for all encrypted files</td>
<td>a year ago</td>
</tr>
</tbody>
</table>
The current repository was configured using transcrypt version 0.9.6 and has the following configuration:

CIPHER:   aes-256-cbc
PASSWORD: MEu5xyQ&G@/}{D\___1231aala4

Copy and paste the following command to initialize a cloned repository:

transcrypt -c aes-256-cbc -p 'MEu5xyQ&G@/}{D\___1231aala4'
Transparent file encryption in git https://www.agwa.name/projects/git-crypt/

Branch: master  New pull request  New file  Upload files  Find file  HTTPS  https://github.com/AGWA/g  Download ZIP

Latest commit b47176e on Dec 26, 2015

- doc
  Add documentation for multiple keys
- man
  Prepare 0.5.0 release
- .gitattributes
  Add .gitattributes file to ignore .git files when creating archive
- .gitignore
  Initial version
- AUTHORS
  Add AUTHORS file
- CONTRIBUTING.md
  Add CONTRIBUTING and THANKS files
- COPYING
  Add README and copyright notices
- INSTALL
  Makefile: refine man page rules
- INSTALL.md
  Makefile: refine man page rules
- Makefile
  Remove gnulib from Makefile
What do you do when a developer leaves the team or the repository is compromised?
VAULT
A tool for managing secrets.

Get Started
Launch Interactive Tutorial
I need access to production DB to fix corrupted data.

service II

access token XYZ leased secret

VAULT

bank integration mysql aws cassandra
I need access to production DB to fix corrupted data.

temporary access token

service II

access token XYZ

Vault

- bank integration
- mysql
- aws
- cassandra

leased secret

leased secret

hard secret
Our recommended approach to use Vault with any configuration manage tool is to move the secret retrieval and renewal into a runtime process instead of a build time process.

- excerpt from Vault documentation
Dynamic infrastructure
Service discovery and configuration made easy. Distributed, highly available, and datacenter-aware.

**Service Discovery**
Consul makes it simple for services to register themselves and to discover other services via a DNS or HTTP interface. Register external services such as SaaS providers as well.

**Failure Detection**
Pairing service discovery with health checking prevents routing requests to unhealthy hosts and enables services to easily provide circuit breakers.
http PUT <consul-url>/v1/catalog/register

NGINX

Service 1
(1 app)

Service 2
(2 apps)

Service 3
(1 app)

Service 4
(3 apps)

Service 5
(5 apps)

consul-template
- modify

nginx.conf

nginx.conf

Makflow

renew

http

upstream app {
  server 1.app.1;
  server 2.app.1;
  server 3.app.1;
  server 4.app.1;
  server 5.app.1;
}

(1 "Address": "1.app.1",
"Service": "app",
"Service": "app",
"Port": 80
)
http PUT <consul-url>/v1/catalog/register

Service 1
(1 app)

Service 2
(2 apps)

Service 3
(3 apps)

Vault

NGINX

Generate

nginx.conf
http {  
  $http_prefix = $app_id $app_id;  
  server {  
    listen 80;  
    server_name $http_prefix.$domain;  
    location / {  
      proxy_pass http://$upstream_addr;  
      proxy_set_header Host $host;  
      proxy_set_header X-Real-IP $remote_addr;  
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
    }  
  }  
}

postgresql

database

application.yml
connection: 
  host: $host
  port: $port
  user: $user
  password: $password

amazon web services
Note that this definition of “config” does not include internal application config, such as `config/routes.rb` in Rails, or how code modules are connected in Spring. This type of config does not vary between deploys, and so is best done in the code.

Another approach to config is the use of config files which are not checked into revision control, such as `config/database.yml` in Rails. This is a huge improvement over using constants which are checked into the code repo, but still has weaknesses: it’s easy to mistakenly check in a config file to the repo; there is a tendency for config files to be scattered about in different places and different formats, making it hard to see and manage all the config in one place. Further, these formats tend to be language- or framework-specific.

**The twelve-factor app stores config in environment variables** (often shortened to `env vars` or `env`). Env vars are easy to change between deploys without changing any code; unlike config files, there is little chance of them being checked into the code repo accidentally; and unlike custom config files, or other config mechanisms such as Java System Properties, they are a language- and OS-agnostic standard.

Another aspect of config management is grouping. Sometimes apps batch config into named groups (often called “environments”) named after specific deploys, such as the `development`, `test`, and `production` environments in Rails. This method does not scale cleanly: as more deploys of the app are created, new environment names are necessary, such as `staging` or `qa`. As the project grows further, developers may add their own special environments like `joes-staging`, resulting in a combinatorial explosion of config which makes managing deploys of the app very brittle.

In a twelve-factor app, env vars are granular controls, each fully orthogonal to other env vars. They are never grouped together as “environments”, but instead are independently managed for each deploy. This is a model that scales up smoothly as the app naturally expands into more deploys over its lifetime.
Reliable deployments require Reliable Deployment System
All changes to Jenkins go through version control and Jenkins can be rebuilt in automated fashion
Golden Image
def project = 'quidryan/aws-sdk-test'
def branchApi = new URL("https://api.github.com/repos/${project}/branches")
def branches = new groovy.json.JsonSlurper().parse(branchApi.newReader())
branches.each {
    def branchName = it.name
    def jobName = "${project}-${branchName}".replaceAll('/','-')
    job(jobName) {
        scm {
            git("git://github.com/${project}.git", branchName)
        }
        steps {
            maven("test -Dproject.name=${project}/${branchName}")
        }
    }
}
gradle-jenkins-plugin

Gradle plugin to programmatically configure Jenkins jobs. This plugin allows you to maintain jenkins job configurations in source control and apply them to the server via gradle. Jobs can be stored as straight xml files, xml strings, markup builder closures, or jenkins job dsl. Job templates can be defined that can then be manipulated such that multiple jobs can be generated off of a single template definition.

See https://github.com/ghale/gradle-jenkins-plugin/wiki for details on usage.
TTD and TTR metrics are defined by SLA of the most critical system
Metadata survives the crash (build number, logs, history)
Slaves survive the crash
All changes to Jenkins are pre-flight tested
Each team has their own Jenkins and owns underlying infrastructure
Hardware is never a bottleneck
Jenkins is either auto-scalable or easy to scale

Consider Jenkins EC2 plugin
Jenkins is at the close proximity with dependencies (which are redundant)
On Friday, November 30th, GitHub had a rough day. We experienced 18 minutes of complete unavailability along with sporadic bursts of slow responses and intermittent errors for the entire day. I'm very sorry this happened and I want to take some time to explain what happened, how we responded, and what we're doing to help prevent a similar problem in the future.

Note: I initially forgot to mention that we had a single fileserver pair offline for a large part of the day affecting a small percentage of repositories. This was a side effect of the network problems and their impact on the high-availability clustering between the fileserver nodes. My apologies for missing this on the initial writeup.
On Saturday, December 22nd we had a significant outage and we want to take the time to explain what happened. This was one of the worst outages in the history of GitHub, and it's not at all acceptable to us. I'm very sorry that it happened and our entire team is working hard to prevent similar problems in the future.
On Thursday, January 28, 2016 at 00:23am UTC, we experienced a severe service outage that impacted GitHub.com. We know that any disruption in our service can impact your development workflow, and are truly sorry for the outage. While our engineers are investigating the full scope of the incident, I wanted to quickly share an update on the situation with you.

A brief power disruption at our primary data center caused a cascading failure that impacted several services critical to GitHub.com's operation. While we worked to recover service, GitHub.com was unavailable for two hours and six minutes. Service was fully restored at 02:29am UTC. Last night we completed the final procedure to fully restore our power infrastructure.

Millions of people and businesses depend on GitHub. We know that our community feels the effects of our site going down deeply. We’re actively taking measures to improve our resilience and response time, and will share details from these investigations.
APP SERVER AVAILABILITY
95.5017%

MEAN WEB RESPONSE TIME
98ms

MEAN API RESPONSE TIME
52ms

98TH PERC. WEB RESPONSE TIME
481ms

PAGES BUILDS FAILURE RATE
0.3921%

Trey @_treyx • Mar 21
ouch githubdown github
Web Testing

phantomjs
xvfb
ievms
browsersync
saucelabs / browserstack / aws device farm
mitmproxy / mitmdump / tamper

Stress Testing

gatling / loader.io / flood.io
simian army

Fault Tolerance

risk storming
timeouts
circuit breakers / graceful degradation
bulkheads
handshaking / rate limiting
cloudflare
auto-scaling
continuous security
Consider

zero-time data migrations (flyway, LHM, pt-online-schema-change, testing migrations on replica)

immutable infrastructure (vagrant / vmware / ansible / serverspec / packer)

terraform

serverless

consumer-driven contracts (accurest in particular)
Implementing Continuous Delivery

find a bottleneck, set a goal
find the best simplest possible solution and get sh%t done
repeat.

More
regular devops get-togethers
commitment language
80/20
reduce batching
build quality in
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
Thank you!
Continuous Security
Validating Git for things that look suspicious:

- Gitrob ([https://github.com/michenriksen/gitrob](https://github.com/michenriksen/gitrob))

- Talisman ([https://github.com/thoughtworks/talisman](https://github.com/thoughtworks/talisman))
GAUNTLT
BE MEAN TO YOUR CODE AND LIKE IT

Join the mailing list for the latest updates

Gauntlt provides hooks to a variety of security tools and puts them within reach of security, dev and ops teams to collaborate to build rugged software. It is built to facilitate testing and communication between groups and create actionable tests that can be hooked into your deploy and testing processes.
Scenario: Verify server is open on expected set of port
  When I launch an "nmap" attack with:
    ""
    nmap -F <hostname>
    ""
  Then the output should match:
    ""
    80/tcp\+open
    ""

Scenario: Verify that there are no unexpected ports open
  When I launch an "nmap" attack with:
    ""
    nmap -F <hostname>
    ""
  Then the output should not contain:
    ""
    22/tcp
    25/tcp
    ""
Scenario: Ensure no anonymous certificates

When I launch an "sslyze" attack with:

```
python <sslyze_path> <hostname>:443
```

Then the output should not contain:

```
Anon
```
pathod: pathological HTTP
Crafted malice for tormenting HTTP clients and servers

pathod
A pathological web daemon.

pathoc
A perverse HTTP client.
The Vulnerability Coordination & Bug Bounty Platform
77% of Programs Find Security Vulnerabilities within 24 Hours
Try HackerOne
We help security minded organizations run successful crowdsourced security programs.

Run a Program  I'm a Researcher

or find out why crowdsourced security can help you.

“Their researchers dig deep in their testing. Not only will they take a URL and test it for many days, but they also find what other systems just can’t identify.”

David Levin, Western Union

View All Public Programs
OWASP Application Security Verification Standard Project
OWASP Top 10 - 2013
The Ten Most Critical Web Application Security Risks
release
The OWASP Zed Attack Proxy (ZAP) is one of the world’s most popular free security tools and is actively maintained by hundreds of international volunteers*. It can help you automatically find security vulnerabilities in your web applications while you are developing and testing your applications. It’s also a great tool for experienced pentesters to use for manual security testing.
Example security tests using Selenium WebDriver and OWASP ZAP

<table>
<thead>
<tr>
<th>Branch: master</th>
<th>New pull request</th>
<th>New file</th>
<th>Upload files</th>
<th>Find file</th>
<th>HTTPS</th>
<th>Latest commit 11b64e1 20 days ago</th>
<th>5 months ago</th>
<th>11 months ago</th>
<th>28 days ago</th>
<th>5 months ago</th>
</tr>
</thead>
<tbody>
<tr>
<td>.idea/libraries</td>
<td>Updated zap-api with getHtmlReport</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>drivers</td>
<td>Updated for OWASP ZAP 2.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libs</td>
<td>Updated zap-api with getHtmlReport</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>src</td>
<td>Updated zap-api with getHtmlReport</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>README.md</td>
<td>Updated README</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>license.txt</td>
<td>Create license.txt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pom.xml</td>
<td>Updated zap-api with getHtmlReport</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zap-webdriver.ilm</td>
<td>Updated zap-api with getHtmlReport</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
switch (policyName.toLowerCase()) {
    case "directory-traversal":
        scannerIds = "4";
        break;
    case "cross-site-scripting":
        scannerIds = "501,5005,50006,50017";
        break;
    case "sql-injection":
        scannerIds = "6018";
        break;
    case "path-traversal":
        scannerIds = "7";
        break;
    case "remote-file-inclusion":
        scannerIds = "77";
        break;
    case "server-side-inclusion":
        scannerIds = "7008";
        break;
    case "script-esc-drupal":
        scannerIds = "8000";
        break;
    case "server-side-code-injection":
        scannerIds = "80016";
        break;
    case "robots-oswald-injection":
        scannerIds = "90001";
        break;
    case "external-servlet":
        scannerIds = "20019";
        break;
    case "xref-injection":
        scannerIds = "20003";
        break;
    case "source-code-disclosure":
        scannerIds = "41,60045,60017";
        break;
    case "null-trick":
        scannerIds = "10008";
        break;
    case "robots-core-execution":
        scannerIds = "20018";
        break;
    case "tom-injection":
        scannerIds = "20015";
        break;
    case "south-injection":
        scannerIds = "50012";
        break;
    case "anti-external-entropy":
        scannerIds = "80023";
        break;
    case "padding-ordinals":
        scannerIds = "90001";
        break;
    case "el-injection":
        scannerIds = "90012";
        break;
}
Evil user stories

As {some kind of bad guy} I want to {do some bad thing}...