CI/CD and TDD in deploying Kamailio

Aleksandar Sošić
@alexsosic
KWC 2018 - Berlin

Who am I?

I’m a DevOp

Owner of kinetic.hr

Developing web applications for more than a decade

R&D manager at a SIP/VoIP cloud startup

Passionate about the Lean Startup methodology, IoT, Blockchain and Cryptocurrencies

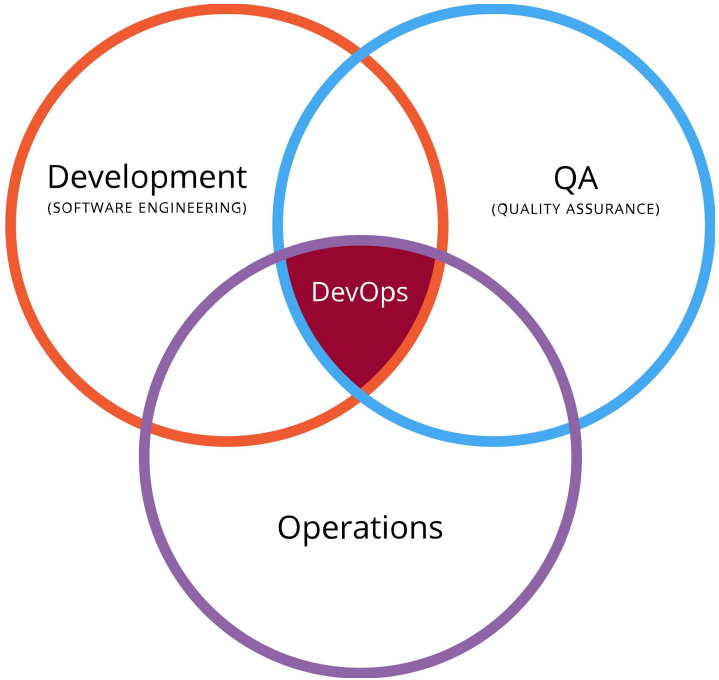
Outdoor enthusiast, photographer and alpinist

Introduction

- Usage of the DevOps cross-functional mode of working with CI/CD in a complex multilayer microservice SIP infrastructure
- Testing single routes in Kamailio on different layers of the architecture
- Test automation with Jenkins in a containerized environment
- Test-driven Development of Kamailio routes

Introductory concepts

DevOps



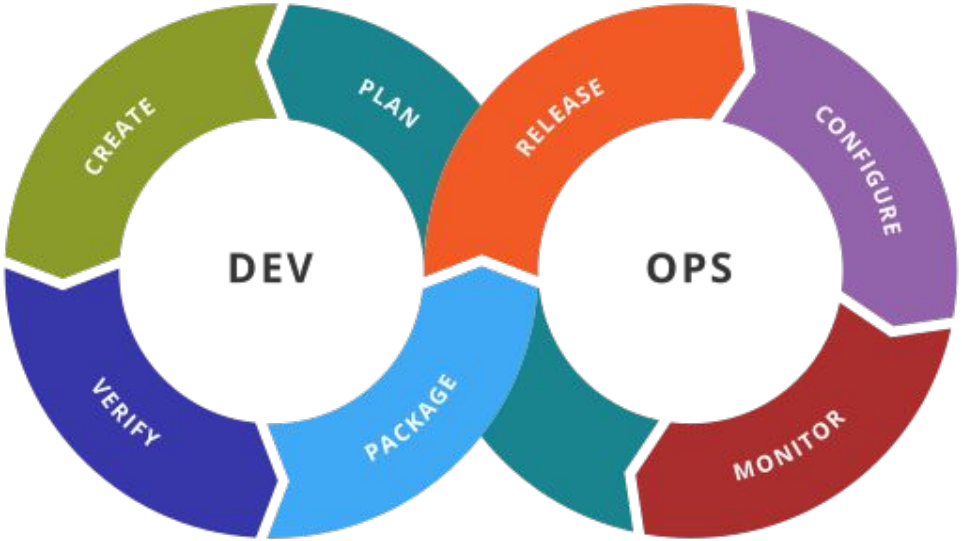
Practice that unifies software development (Dev) and software operation (Ops)

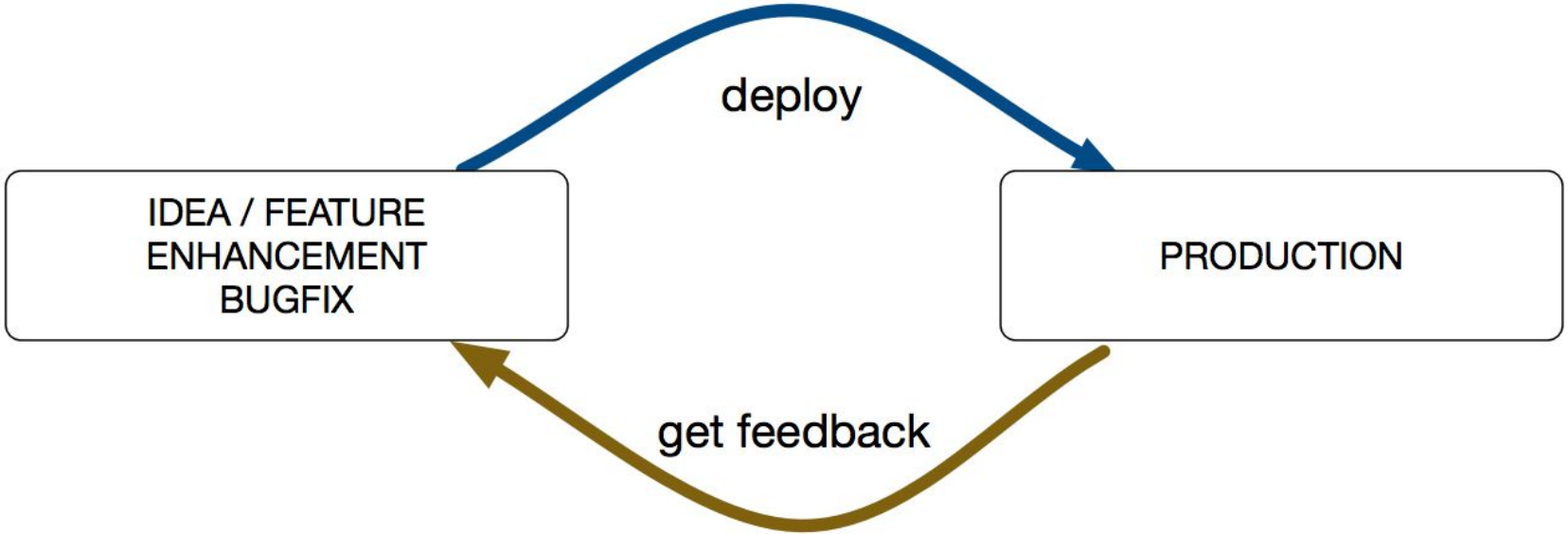
- Shorter development cycles
- increased deployment frequency
- more dependable releases in close alignment with business objectives

Automation and monitoring of all steps is the key!

DevOps

DevOps is intended to be a cross-functional mode of working




deploy
IDEA / FEATURE
ENHANCEMENT
BUGFIX
PRODUCTION
get feedback

Goals of DevOps

- Improved deployment frequency
- Faster time to market
- Lower failure rate of new releases
- Shortened lead time between fixes
- Faster mean time to recovery

CI/CD

Continuous integration (CI) is the practice of merging all developer working copies to a shared mainline several times a day.

The main goal is to prevent integration problems, referred to as "integration hell".

CI/CD

Continuous delivery (CD) is an approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time.

Relationship to continuous deployment?

Continuous delivery and DevOps have common goals and are often used in conjunction

Microservice Infrastructure

AKA **microservice architecture**

- Structures an application as a collection of loosely coupled services
- Enables the continuous delivery/deployment of large, complex applications
- Enables an organization to evolve its technology stack
- Services are fine-grained and the protocols are lightweight
- Improves modularity
- Makes the application easier to understand, develop and test
- Enable continuous delivery and deployment

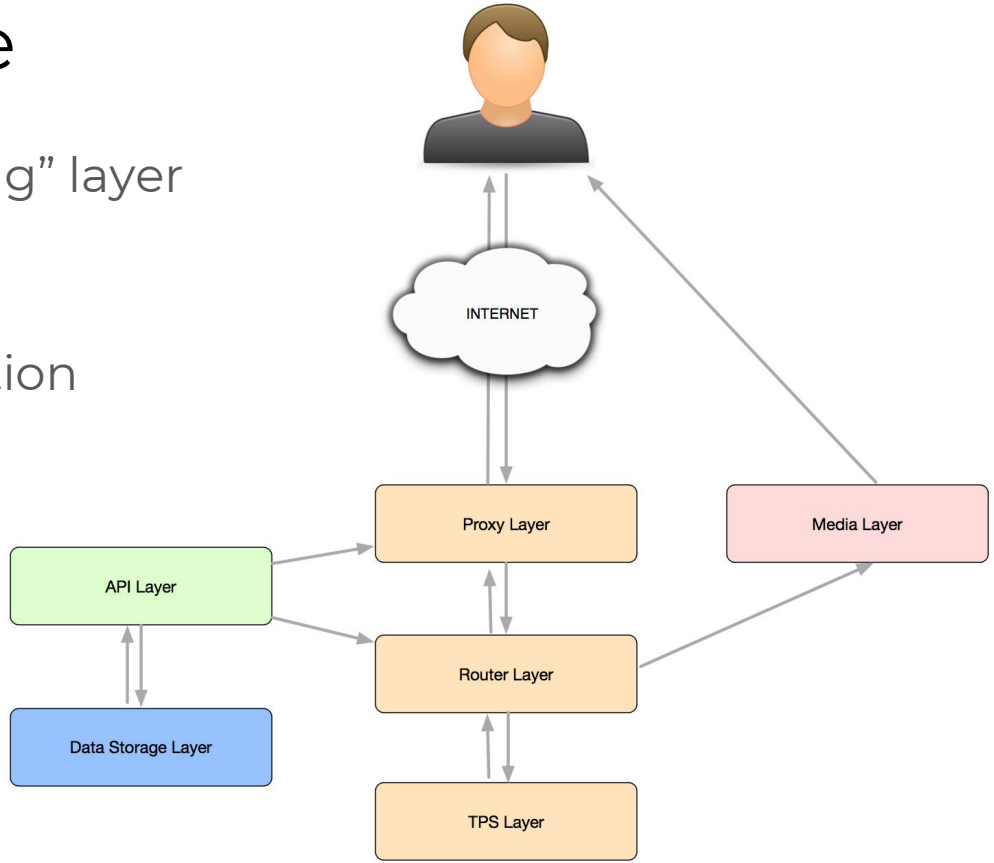
Our project architecture

Project features and technologies used

- Containerized environment
- Stateless Kamailio instances
- Multi layer infrastructure
- K8S as container orchestrator
- Custom API for SIP infrastructure orchestration
- External Clustered DB

Project architecture

- Kamailio proxy and “routing” layer
- Asterisk as TPS
- RTPEngine
- Custom API SIP orchestration



Proxy and Router roles

Proxy:

- Proxy load balancing (callid path maintained)
- First security layer (ua/pike)

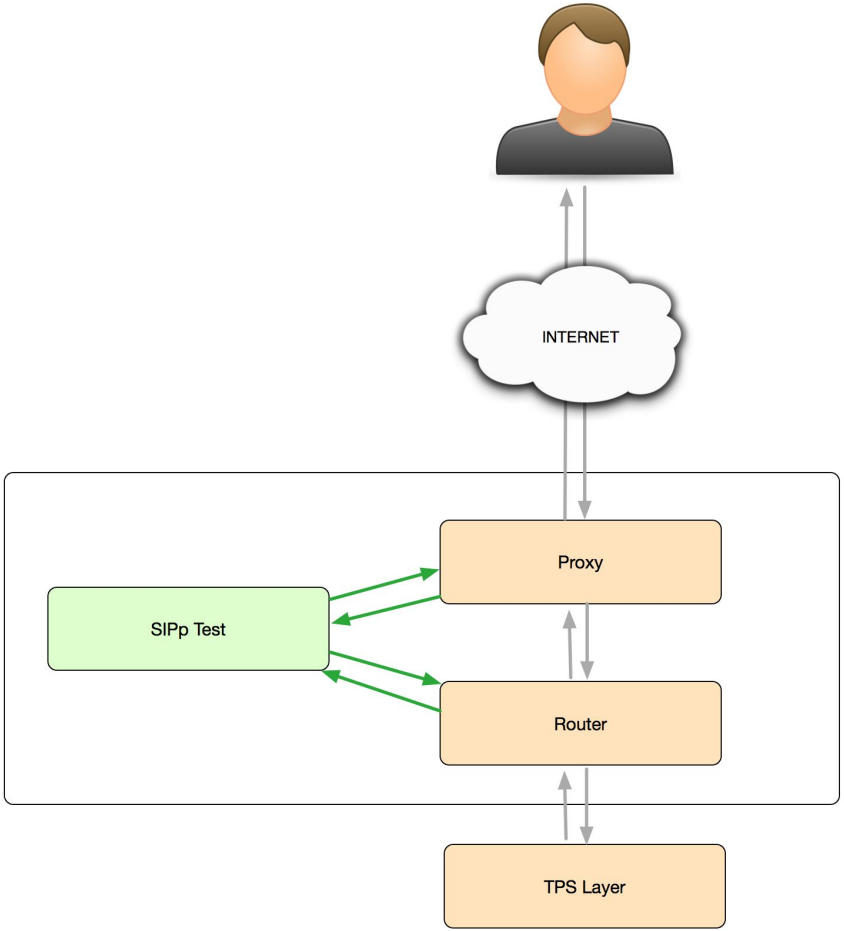
Router:

- User registration & management
- Session control
- Security (user authentication/ip filtering etc.)
- Billing
- Number normalization
- E2E routing (LCR & Carrier / Failover & Re-routing)

Testing Kamailio Routes

kamailio.cfg

```
#!ifdef TESTING
include_file "kamailio-test.cfg"
#!endif
...
request_route {
#!ifdef TESTING
  if ($hdr(X-evosip-Test) =~ "^TEST_") {
    route($(hdr(X-evosip-Test){s.rm,"}));
    exit;
  }
#!endif
...
```


INTERNET
Proxy
SIPp Test
Router
TPS Layer

SIPp

Free Open Source test tool traffic generator for the SIP protocol

- establishes and releases multiple calls
- custom XML scenario files describing call flows
- dynamic display of statistics about running tests (call rate, round trip delay, and message statistics)
- periodic CSV statistics dumps

SIPp xml scenario

```
<send retrans="500">
  <![CDATA[
    OPTIONS sip:[field2]@[remote_ip]:[remote_port] SIP/2.0
    Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
    From: "sipvicious" <sip:[field0]@[field1]>;tag=[call_number]
    User-Agent: sipvicious
    To: "sipvicious" <sip:[field0]@[field1]:[remote_port]>
    X-evosip-Test: TEST_IS_SCANNER
    Call-ID: [call_id]
    CSeq: [cseq] OPTIONS
    Contact: sip:[field3]@[local_ip]:[local_port]
    Max-Forwards: 70
    Content-Type: application/sdp
    Content-Length: [len]
  ]]>
</send>
<recv response="222" crlf="true"></recv>
```

Testing Kamailio

```
route[TEST_IS_SCANNER] {
  if(route(IS_SCANNER)) {
    xlog("L_WARNING", "SCANNER $ua BLOCKED!\n");
    sl_send_reply("222",
       "Test passed, $ua scanner blocked!");
  }
  else {
    sl_send_reply("500", "Test failed: NOT A SCANNER");
  }
}
```

Testing Kamailio

```
#!define SCNR_BLACK_LIST "sipsak|sipvicious|..."
...
route[IS_SCANNER] {
  if ($ua =~ SCNR_BLACK_LIST) {
    return 1;
  } else {
    return 0;
  }
}
```

Writing more complex tests

Dynamic dispatcher list example

Route SHOULD_DISPATCHER_RELOAD

- Are the dispatchers already reloaded?
- Am I reloading the dispatchers?
- Does it take too long to reload the dispatchers?

Writing more complex tests

```
route[SHOULD_DISPATCHER_RELOAD] {
  if($sht(dispatcher=>list) == 1) {
    return 0;
  } else {
    $var(todate) = $(sht(dispatcher=>list){s.int}) + 60;
    if ($var(todate) < $TV(sn)) {
      return 1;
    } else {
      return 0;
    }
  }
}
```

Writing more complex tests

```
route[TEST_SHOULD_DISPATCHER_RELOAD] {
    $var(TestsPassed) = 0;
    $var(TestsNum) = 3;

    # Dispatcher list has been correctly reloaded
    # and should not be reloaded again
    $sht(dispatcher=>list) = 1;
    if(!route(SHOULD_DISPATCHER_RELOAD)) {
        $var(TestsPassed) = $var(TestsPassed) + 1;
    }
...
```

Writing more complex tests

```
...
  # The dispatcher list route
  # has been just triggered
  # and should not be called again for 60 seconds
  $sht(dispatcher=>list) = $TV(sn);
  if(!route(SHOULD_DISPATCHER_RELOAD)) {
    $var(TestsPassed) = $var(TestsPassed) + 1;
  }
...
```

Writing more complex tests

```
...
  # The dispatcher list route has been called
  # more than 60 seconds ago and should reload
  $sht(dispatcher=>list) =
    $(sht(dispatcher=>list){s.int}) - 65;
  if(route(SHOULD_DISPATCHER_RELOAD)) {
    $var(TestsPassed) = $var(TestsPassed) + 1;
  }
...
```

Writing more complex tests

```
...
  # Tests concluded count test number
  # and respond via sl_send_reply
  if($var(TestsPassed) == $var(TestsNum)) {
     sl_send_reply("200", "DISPATCHER RELOAD
                    TRIGGER WORKING");
  } else {
    sl_send_reply("500", "DISPATCHER RELOAD
                   TRIGGER NOT WORKING");
  }
}
```

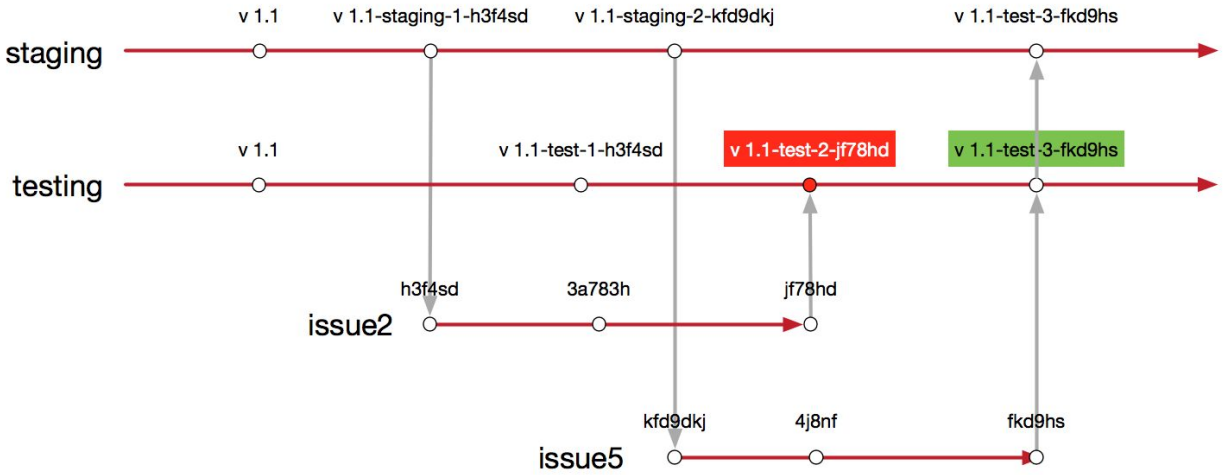
Automate Tests

Jenkins

Jenkins is an open source automation server written in Java.

- Helps us automate the non-human part of the software development process
- Does continuous integration
- Facilitates technical aspects of continuous delivery

CI/CD Flow


v 1.1
v 1.1-staging-1-h3f4sd
v 1.1-staging-2-kfd9dkj
v 1.1-test-3-fkd9hs
staging
v 1.1
v 1.1-test-1-h3f4sd
v 1.1-test-2-jf78hd
v 1.1-test-3-fkd9hs
testing
h3f4sd
3a783h
jf78hd
issue2
kfd9dkj
4j8nf
fkd9hs
issue5

SIPp script for jenkins integration

```
#!/bin/bash

NAME="ut_kama_proxy-scanners"
SIPP=/usr/local/bin/sipp
SCENARIOS=/sipp/scenarios/"${NAME}".xml
SETTINGS=/tmp/inf_files/"${NAME}".csv
DATE_FILE=$(date '+%Y-%m-%d_%H-%M')
NODE_OWNER=$(echo "${POD_NODE}" | cut -d '.' -f 2)
NODE_DOMAIN=$(echo "${POD_NODE}" | cut -d '.' -f3-)
TARGET="proxy.${NODE_OWNER}.${NODE_DOMAIN}"

"${SIPP}" "${TARGET}" -sf "${SCENARIOS}" -l 1 -m 1 -inf "${SETTINGS}"
exit "${?}"
```

Automation with Jenkins

```
stage('functional tests') {
    def testPods = getPodsByPrefix("test-")
    if(testPods.size() < 1) {
      throw new Exception("No test pods found")
    }
    def testPod = testPods[0]
    echo "----\ntest pod: ${testPod.name}\n----"
    echo "executing test 'proxy scanners' on pod ${testPod.name}"
    podExec(testPod, "/sipp/scripts/ut_kama_proxy-scanners.sh", false)
    updateGitlabCommitStatus name: 'func tests', state: 'success'
}
```

Test-driven development

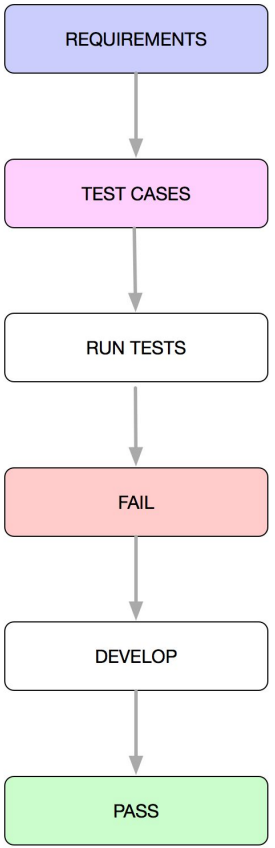
Test-driven development

Test-driven development is related to the test-first programming concepts of extreme programming

TDD relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the software is improved to pass the new tests, only

“TDD encourages simple designs and inspires confidence”

-- Kent Beck



TDD Example

```
route[TEST_IS_SCANNER] {
  if(route(IS_SCANNER)) {
    xlog("L_WARNING", "SCANNER $ua BLOCKED!\n");
    sl_send_reply("222",
       "Test passed, $ua scanner blocked!");
  }
  else {
    sl_send_reply("500", "Test failed: NOT A SCANNER");
  }
}
```

TDD Example

```
<send retrans="500">
  <![CDATA[
    ...
    User-Agent: sipvicious
    ...
    X-evosip-Test: TEST_IS_SCANNER
    ...
  ]]>
</send>
<recv response="222" crlf="true"></recv>
```

TDD Example

```
route[IS_SCANNER] {
  return 0;
}
```

TDD Example

```
route[IS_SCANNER] {
  if ($ua == "sipvicious") {
    return 1;
  } else {
    return 0;
  }
}
```

TDD Example

```
#!define SCNR_BLACK_LIST "sipsak|sipvicious|..."
...
route[IS_SCANNER] {
  if ($ua =~ SCNR_BLACK_LIST) {
    return 1;
  } else {
    return 0;
  }
}
```

Conclusions

- Custom test routes in Kamailio
- Request route condition for access to those routes in a testing environment
- Automation with SIPp scenarios and Jenkins for CI/CD
- Ability to do TDD

www.linkedin.com/in/alexsosic/

alex@kinetic.hr