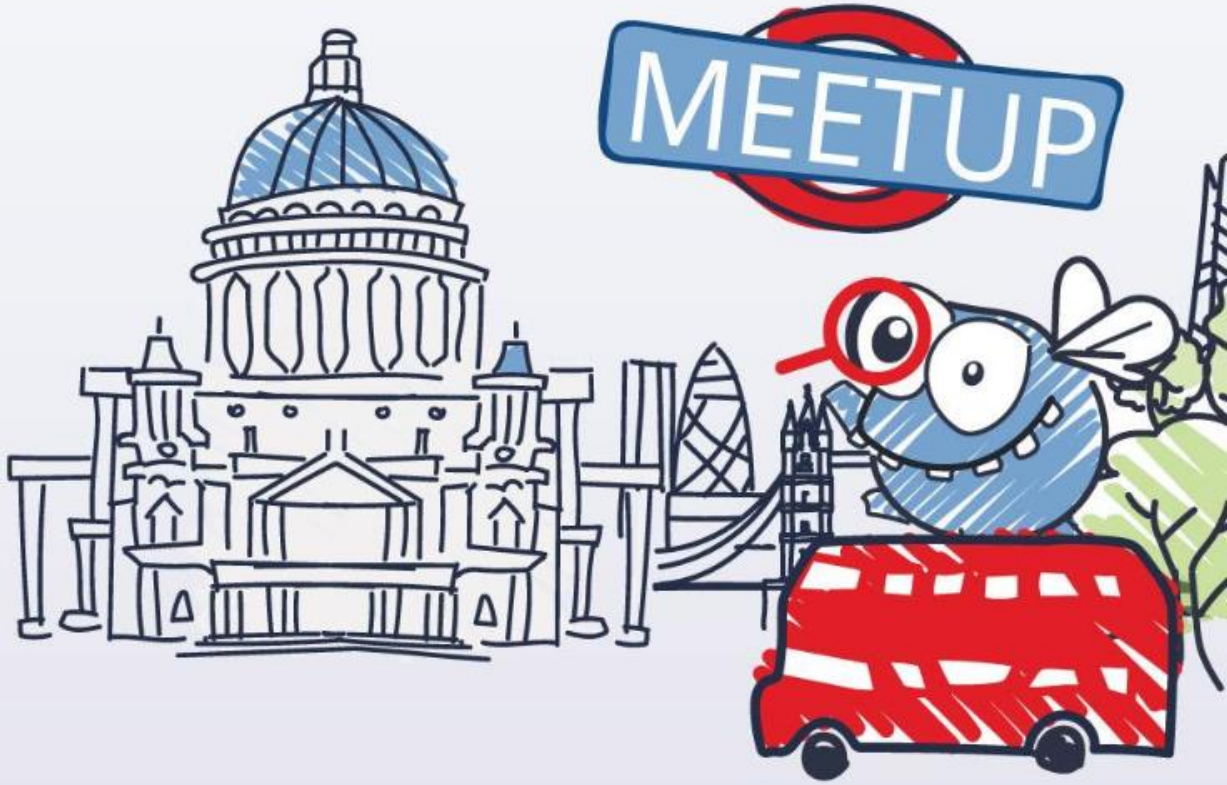
EXTENT CONFERENCE

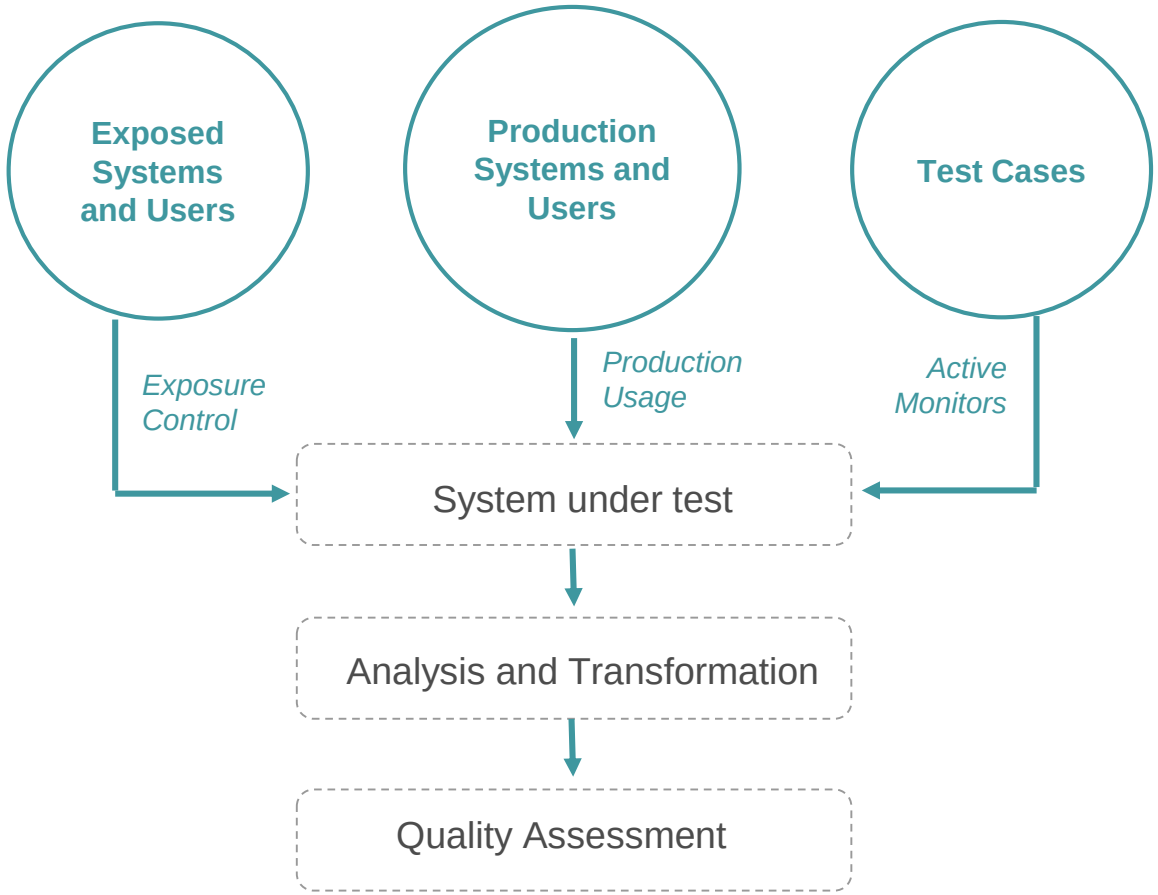
TestOps Environments and Monitoring



Stanislav Klimakov
20th June 2018

What is TestOps?

Use of production as test environment



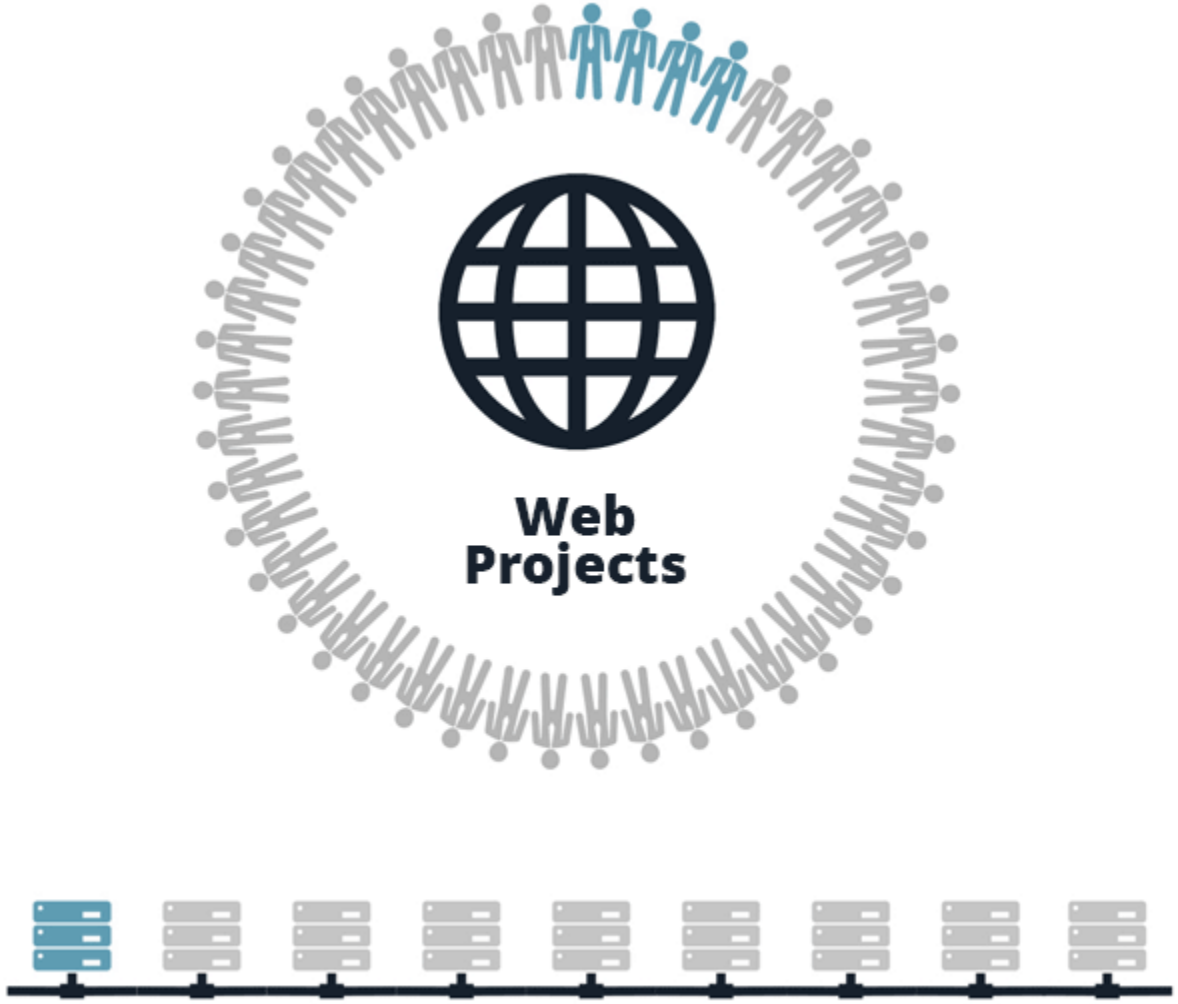
Pros:

- Very fast product delivery cycle
- Wide coverage
- High quality for unexposed end users

Cons:

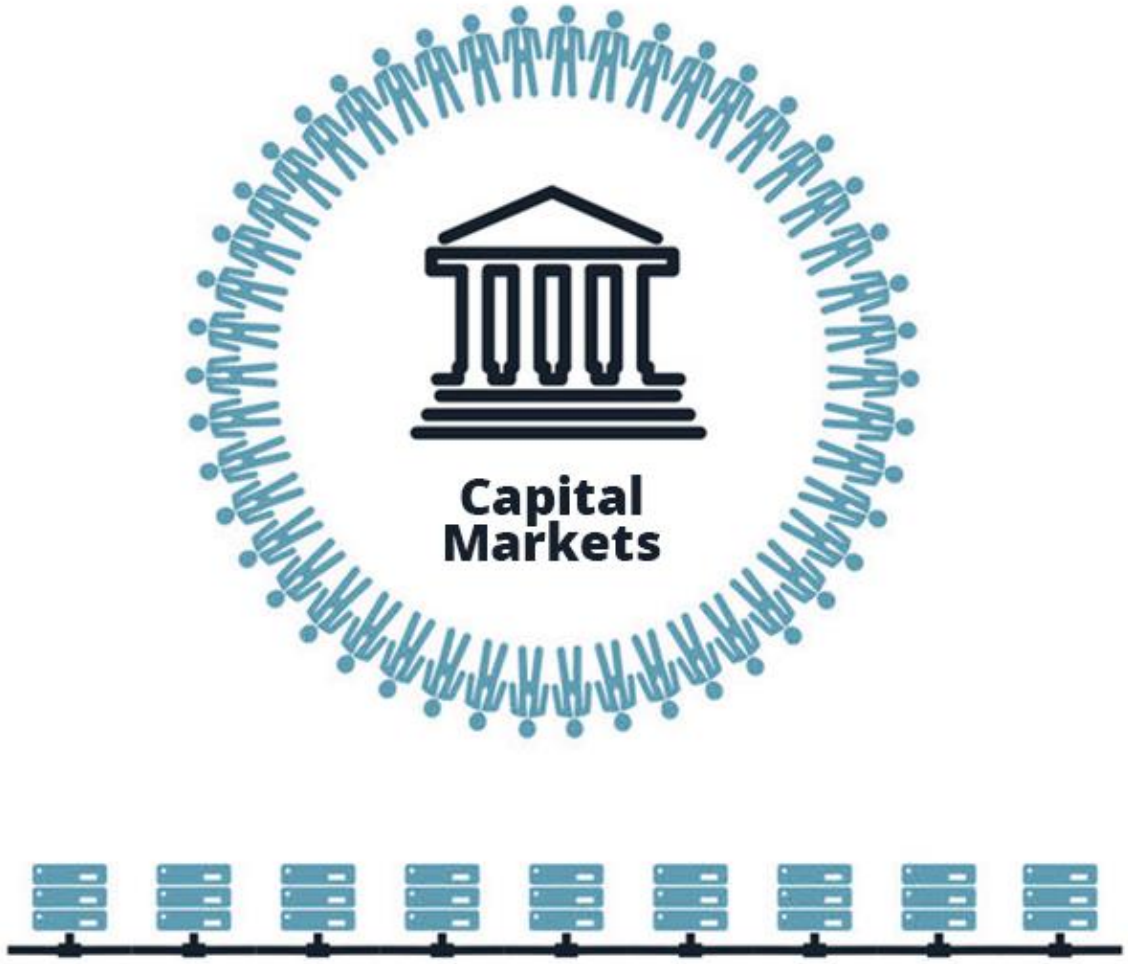
- Exposed users pay for quality
- It is not a solution acceptable in finance

What is TestOps?



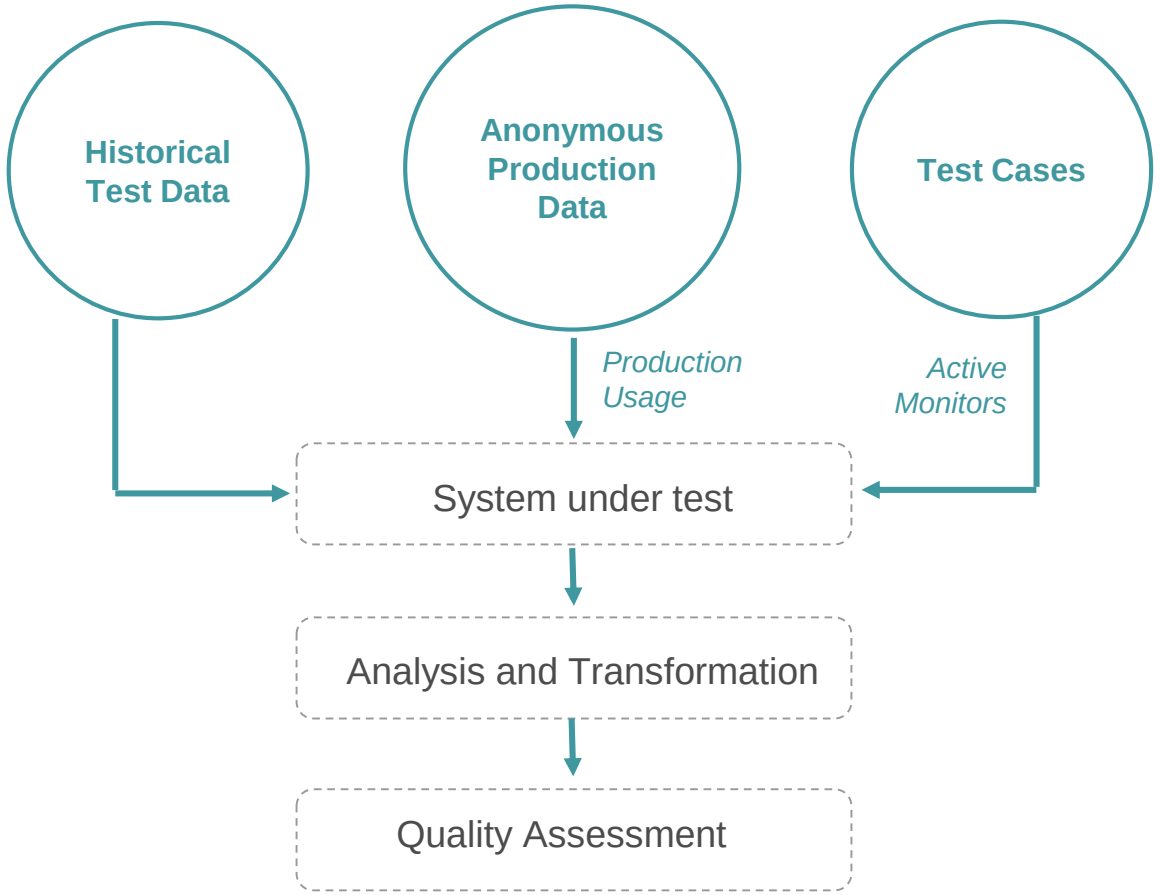
What is TestOps for Us?

What is TestOps for Us?



What is TestOps for Us?

Use of a test environment as a production environment



Pros:

- Fast product delivery cycle
- End users are not exposed

Cons:

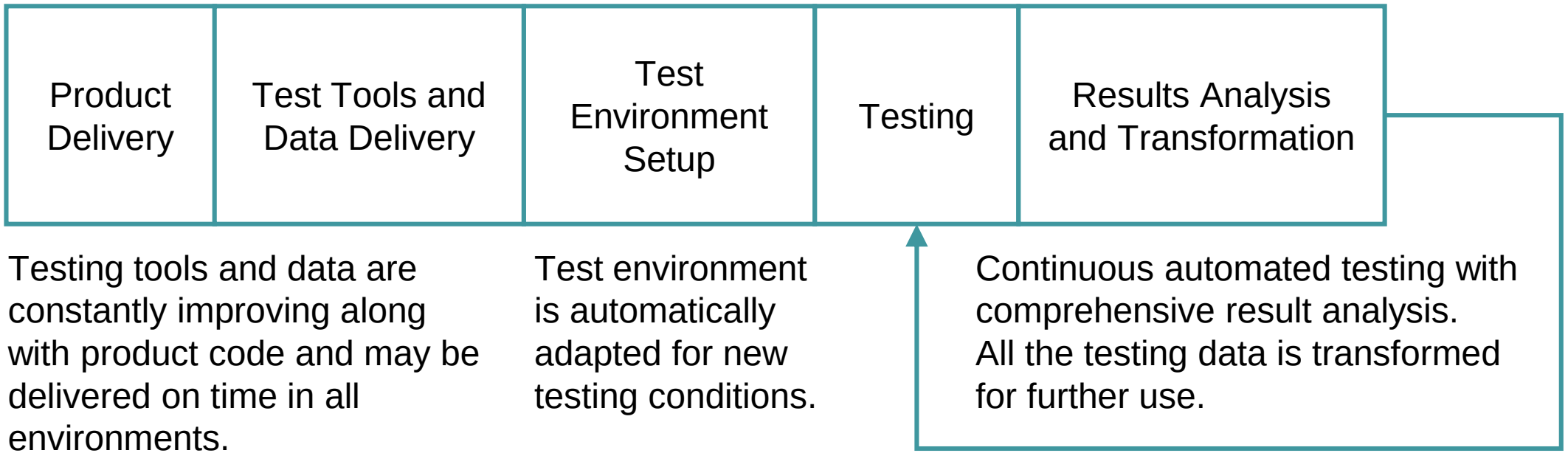
- Poorer coverage in comparison with the “traditional” TestOps approach

What does “production-like” mean?

- Strict configuration and version control
- Restricted access to a limited number components
- Granted availability and stable work during production hours
- Production-only monitoring and operating solutions

- Availability – production-like environment must not be idle and must add value
- Accessibility – the system must grant testers full control
- Additional monitoring

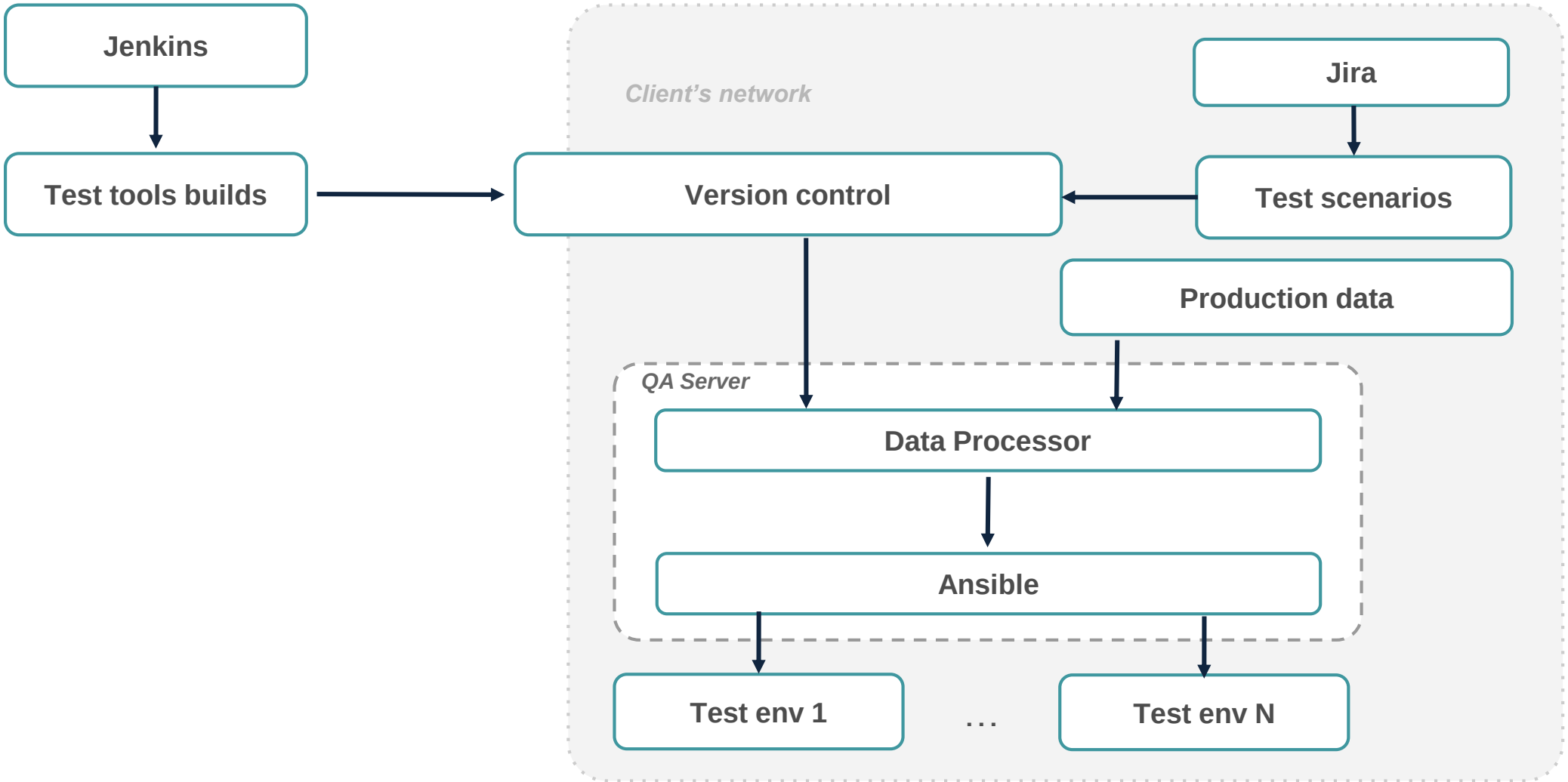
Process Flow



Test Tools Delivery and Environment Setup

- All the tools must be up to date and delivered on time in all environments
- The actual test data must be updated automatically using all accessible information from production
- Environment configuration must be updated gradually when required
- All changes in scenarios must reflect the current environment setup

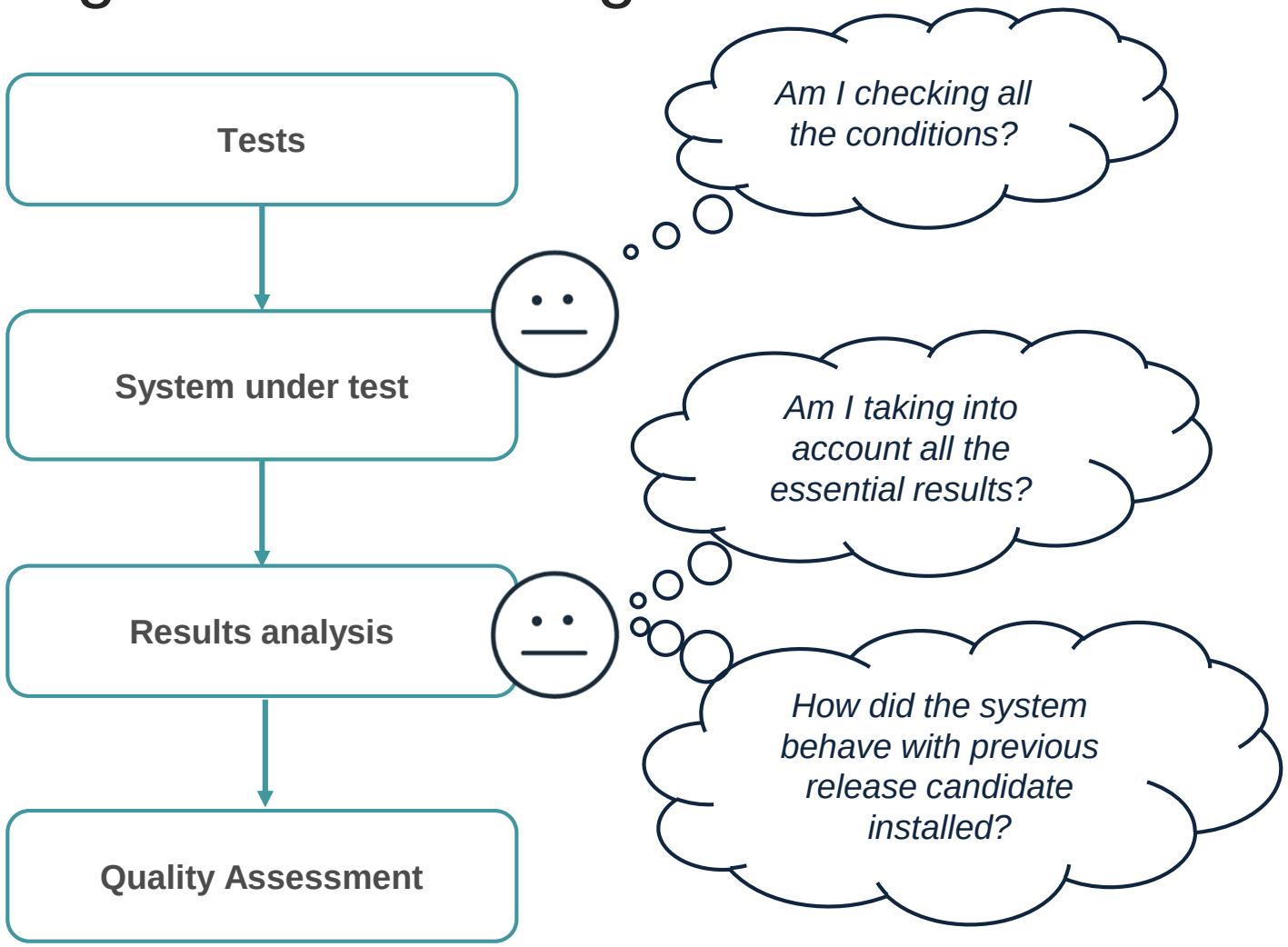
Test Tools Delivery and Environment Setup



Testing

- Tests must be executed 24/7 when possible – an idle system does not help to find issues
- Test execution process must be transparent and user-friendly
- Put efforts into test coverage and improvement, but not test execution

High Touch Testing



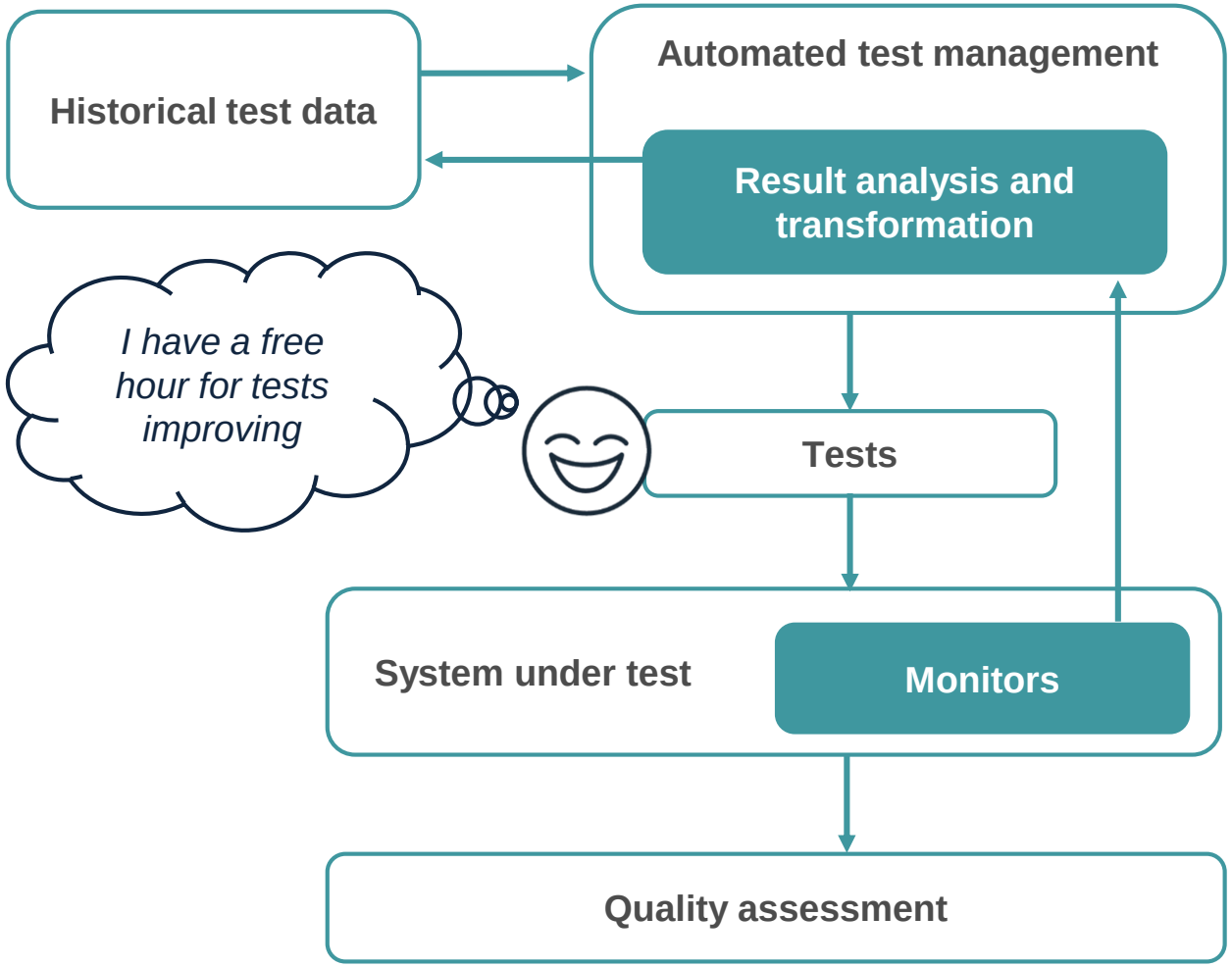
Pros:

- A tester may notice unusual system behavior, like a real system operator

Cons:

- A tester may miss an issue while comparing multiple conditions
- Low efficiency and coverage
- Less time for test improvement

Low Touch Testing



Pros:

- Non-stop test execution
- Less room for a human error
- Less time for analysis
- More time for extending test coverage

Cons:

- Higher tester qualification for improving automated scenarios
- Validators may pass an issue that a tester could have noticed in real time

Automated tests management

EXTENT CONFERENCE

- High level user friendly scenarios
- Event driven automated scenarios execution
- Platform independent and ready to go solution
- Environment specific plugins support

Test scenarios

```
0  start_load
1  kill -9 MatchingEnginePrimary
2  set smoke_status exec_smoke
3  if 'PASS' == #{smoke_status} then goto pass
4  echo FAIL
5  stop_load
6  exit
7  label pass
8  echo PASS
```

```
0  start_load
1  kill -9 MatchingEnginePrimary
2  set smoke_status exec_smoke
3  if 'PASS' == #{smoke_status} then goto pass
8  echo PASS
```

```
0  start_load
1  kill -9 MatchingEnginePrimary
2  set smoke_status exec_smoke
3  if 'PASS' == #{smoke_status} then goto pass
4  echo FAIL
5  stop_load
6  exit
```

Every scenario is language agnostic and consists of a sequence of commands (test steps). All the magic is hidden behind abstracted commands like `exec_smoke` which may be provided by an extension or be just an alias for some system script. Even though smoke test may have different logic in a variety of systems, the main scenario logic remains the same.

Monitoring Network

<u>Purpose:</u>

- Monitor system events along with existing monitoring system provided by vendor

Requirements:

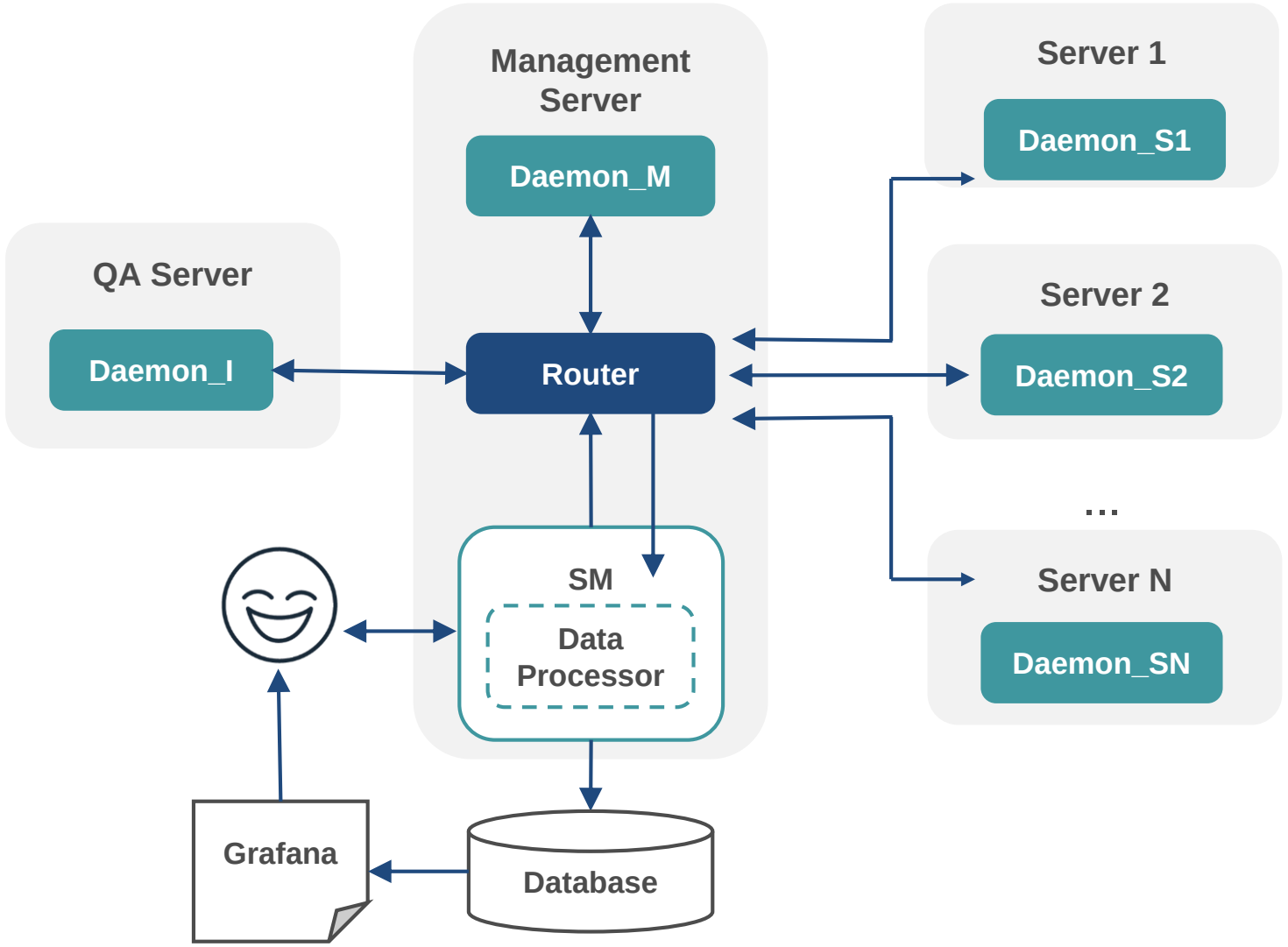
- Standalone tool without 3rd party dependencies for all the tasks
- Easy to control, collect and transform data flows

Why?

- No need to adapt all the environments for our needs
- Automation process requires less effort, all the scripts are standardized

Monitoring Network

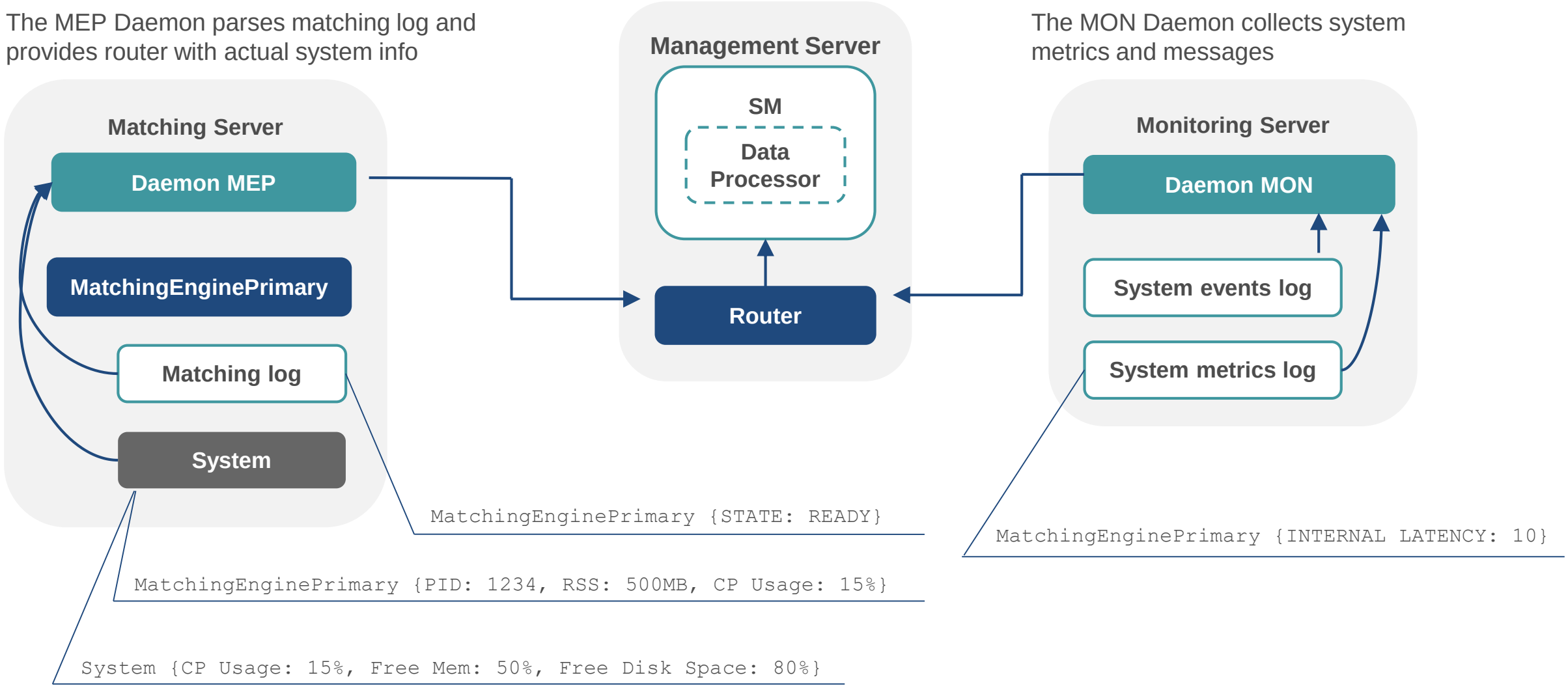
Daemon_S	Collecting system info, logs parsing, commands execution
Daemon_M	Collecting system info, logs parsing, commands execution
Daemon_I	Load control and test scripts execution
Router	Communication between daemons and controllers
SM	Automated execution of test scenarios, collecting and processing test information
Data Processor	Transform, collect and store data for future use
Grafana	Data visualisation



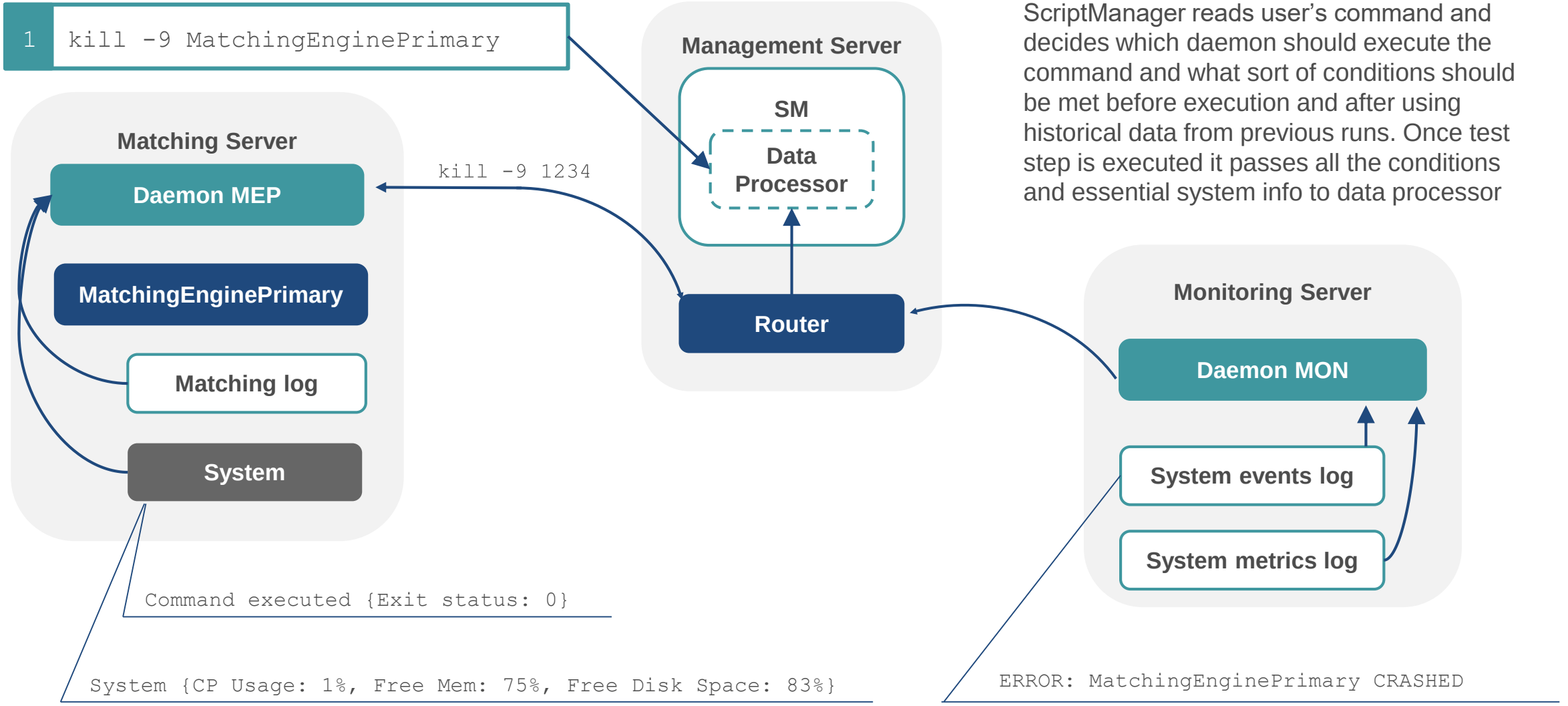
Data Gathering

The MEP Daemon parses matching log and provides router with actual system info

The MON Daemon collects system metrics and messages



Commands Execution



ScriptManager reads user's command and decides which daemon should execute the command and what sort of conditions should be met before execution and after using historical data from previous runs. Once test step is executed it passes all the conditions and essential system info to data processor

Behaviour analysis

Requirements:

- Testing tool cannot completely rely on the deterministic scenario validators while executing tests in a complex distributed environment
- Historical data must be stored and used when “almost the same conditions” are met to compare results with

Problem:

- Exchange operates with thousands internal metrics
- Find what kinds of metrics may be compared in a particular conditions

A solution must rely on AI because no tester is capable of describing all possible combinations.

EXTENT CONFERENCE

Thank you!