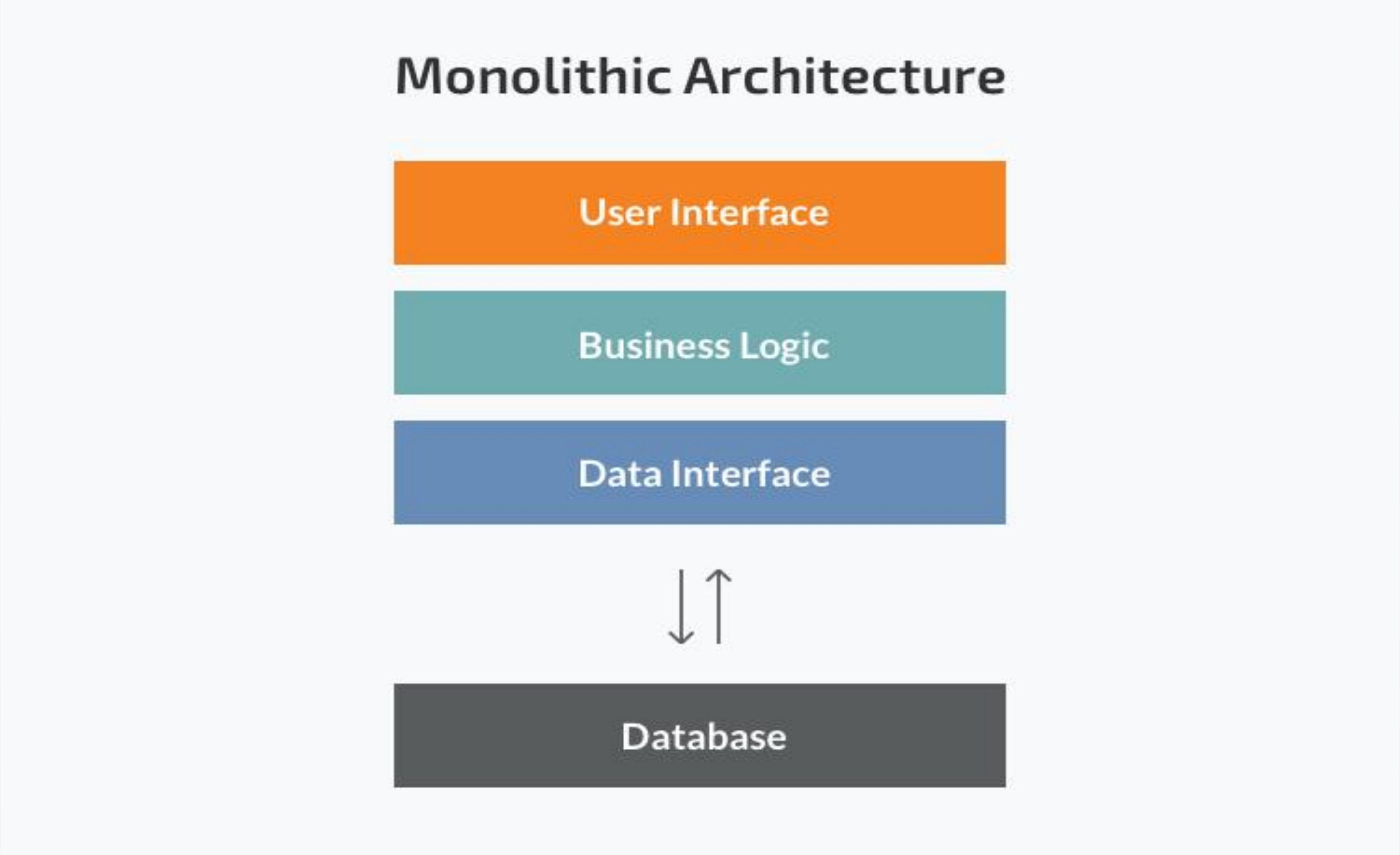
Distributed system technical audit

Agenda

- Distributed systems overview.
    - Monolithic architecture
    - Microservice architecture
    - Serverless architecture
- Technical audit overview.
    - What is technical audit?
    - Quality attributes
- Technical audit checklists

Single deployment unit. Separation of concern was used to manage complexity.



Monolithic architecture

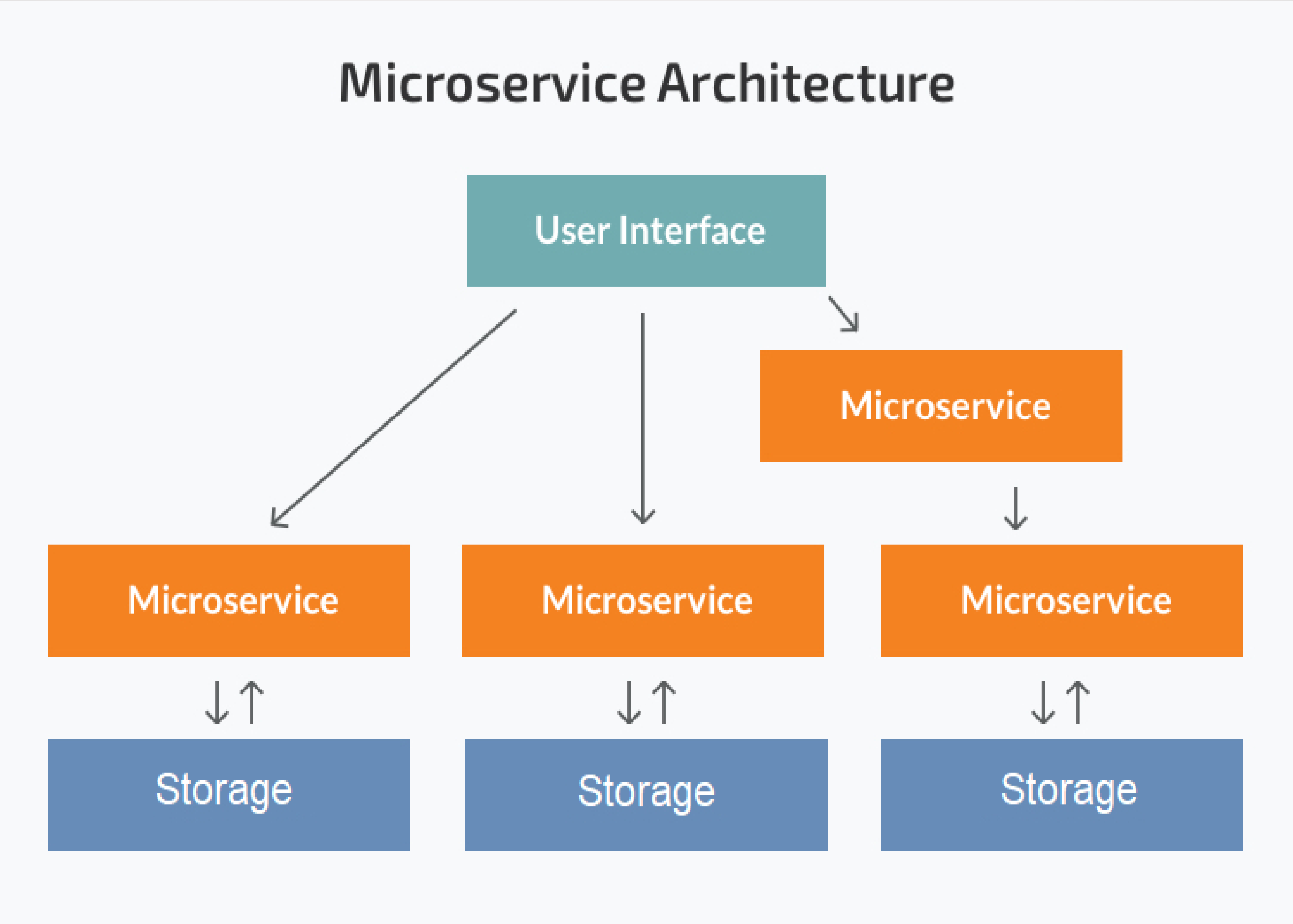
Advantages:

- **Less cross-cutting concerns**
- **Easier debugging and testing**
- **Easier to deploy**

Disadvantages:

- **Understability**
- **Scalability**
- **Updatability**
- **Introduce new technologies**

Microservices architecture breaks single unit into a collection of smaller ones which are not depend on each other.



Microservices architecture

Advantages:

- **Independent deployment units**
- **Better understability**
- **Scalability**
- **Agility**
- **Flexibility in choosing the technology**

Disadvantages:

- **Complexity**
- **Cross-cutting concerns**
- **Testability**

Serverless is a cloud computing execution model where the cloud provider dynamically manages the allocation and provisioning of servers.

Cloud providers

IBM OpenWhisk

AWS Lambda

Azure Functions

Google Cloud Functions

Auth0 Webtask

Serverless architecture

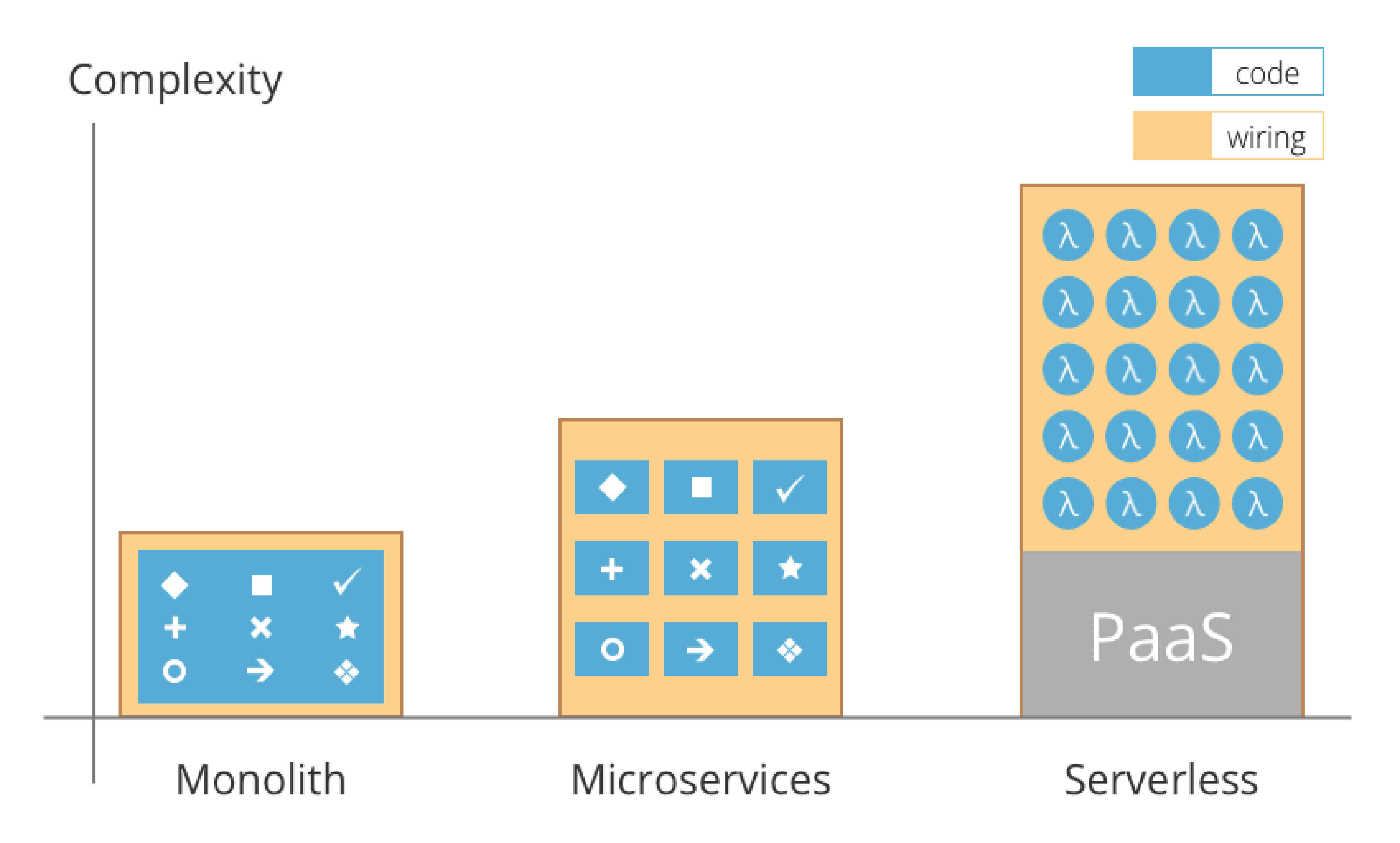
Advantages:

- **Focus on business logic**
- **Scalability**
- **Pay for invocation**
- **Flexibility in choosing the technology**

Disadvantages:

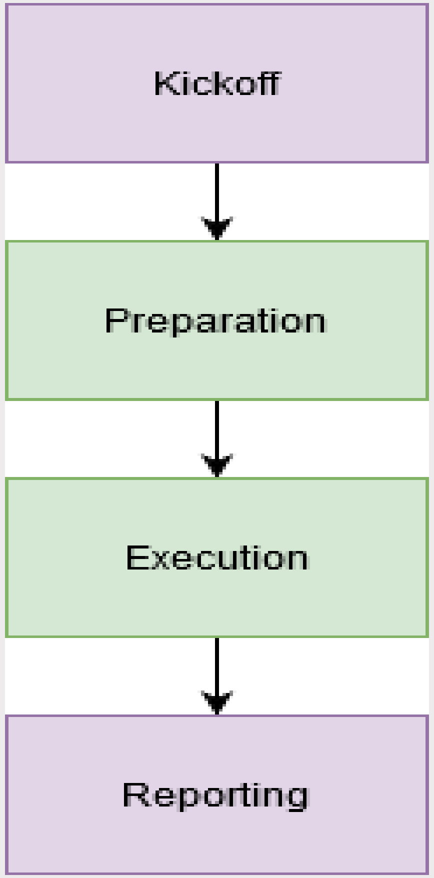
- **Unrelated set of functions**
- **Testability**

When you increase granularity of your system you also increase system complexity.



Audit is a formal procedure to measure a technical debt and a quality level of the system.

Phases of technical audit



Classification of requirements

- Functional requirements
- Non-functional requirements
  - Quality attributes
  - Constraints

Observability is a measure of how well internal states of a system can be inferred from knowledge of its external outputs.

Portability is the ability to deploy a product in various environments in a predictable way.

Security is the ability to resist to incorrect or malicious behavior of client applications.

Maintainability is the ability to change a product with a predictable effort.

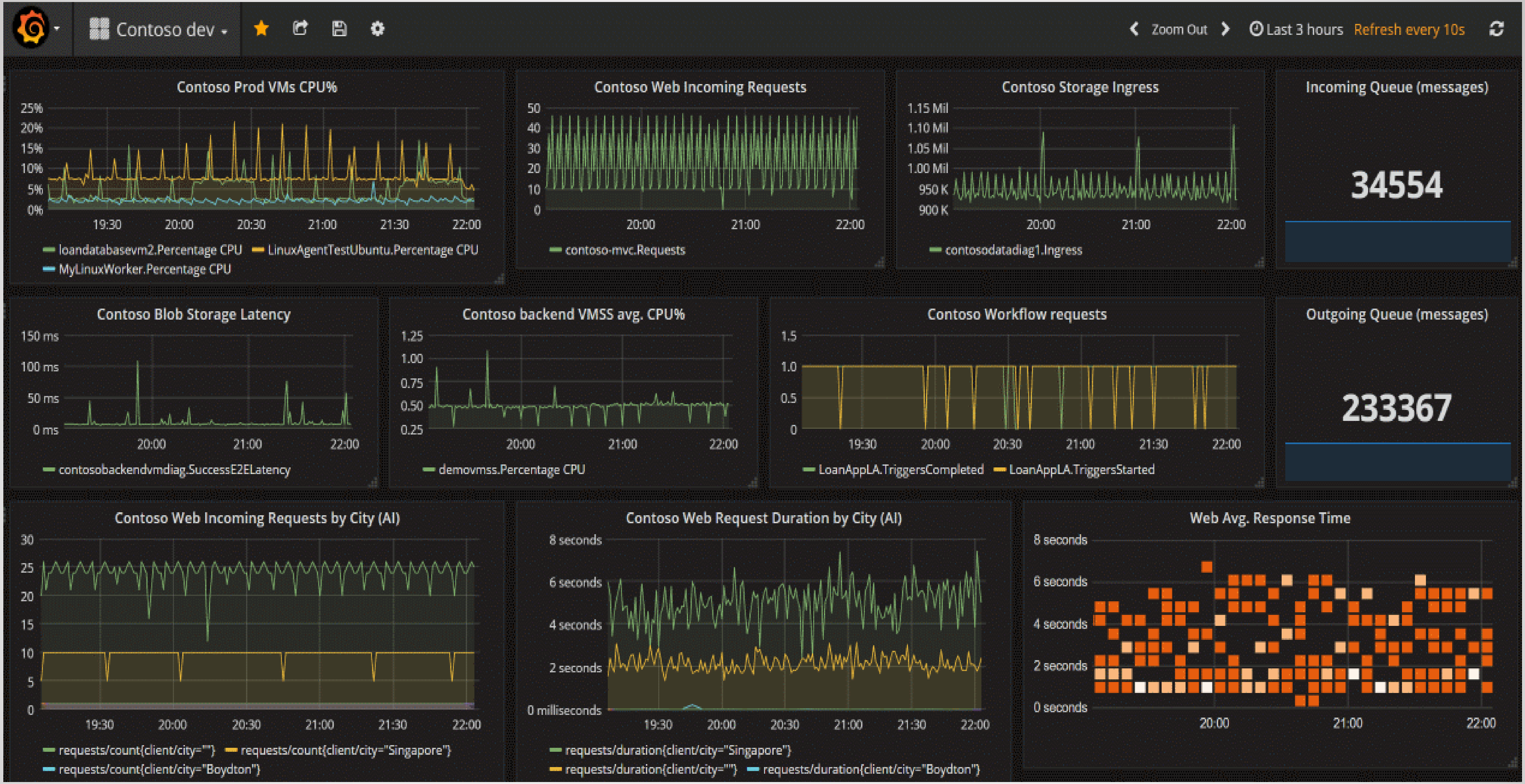
Observability checklist

Must have:

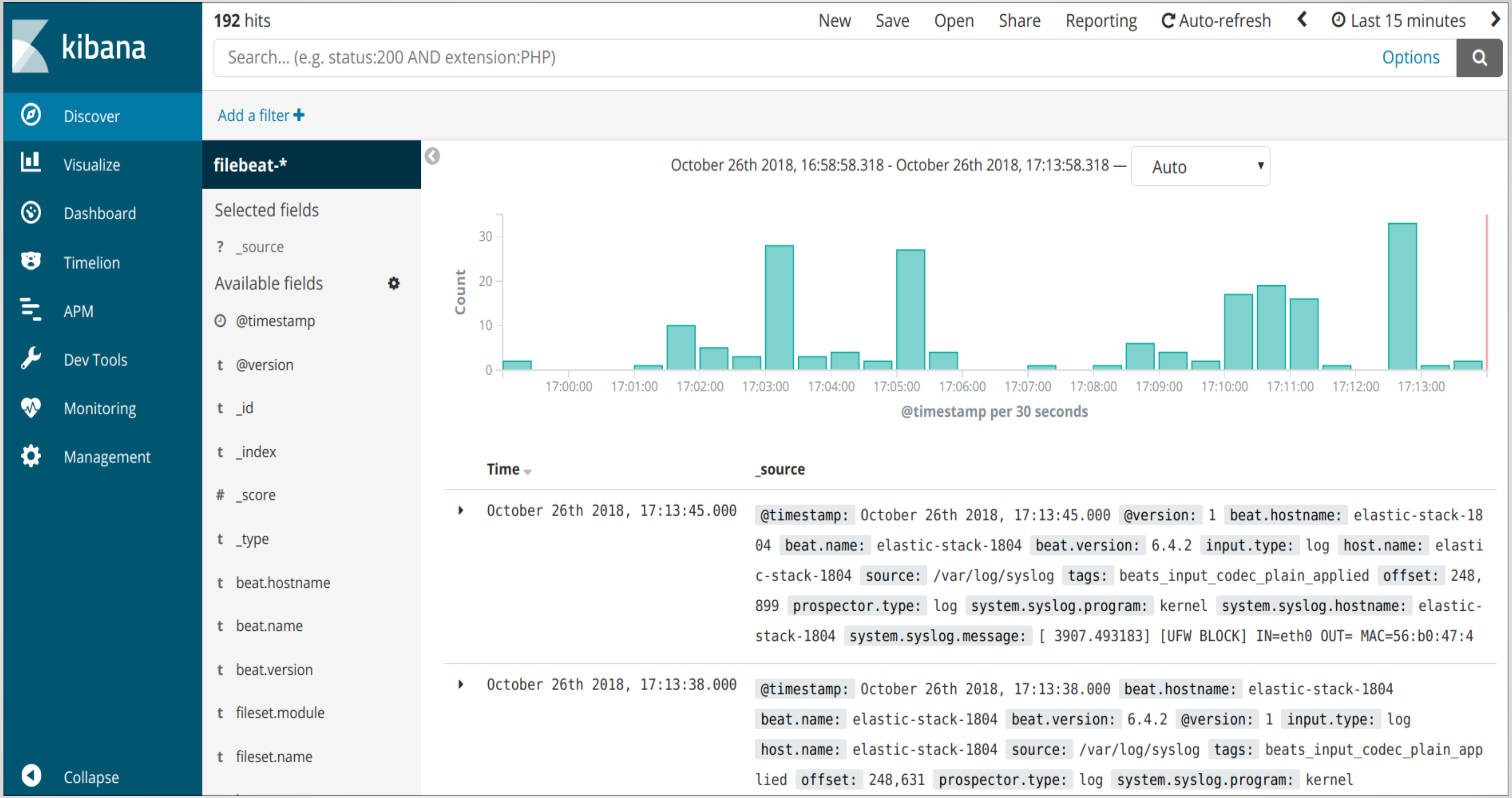
- Use correlations
- Enable logging
- Use log context for instances
- Provide a default error handler
- Use health checks

Should have:

- Enable tracing
- Log context for invocations
- Enable error tracking


Contoso dev
Zoom Out
Last 3 hours
Refresh every 10s
Contoso Prod VMs CPU%
25%
20%
15%
10%
5%
0%
19:30
20:00
20:30
21:00
21:30
22:00
loandatabasevm2.Percentage CPU
LinuxAgentTestUbuntu.Percentage CPU
MyLinuxWorker.Percentage CPU
Contoso Web Incoming Requests
50
40
30
20
10
0
20:00
21:00
22:00
contoso-mvc.Requests
Contoso Storage Ingress
1.15 Mil
1.10 Mil
1.05 Mil
1.00 Mil
950 K
900 K
20:00
21:00
22:00
contosodatadiag1.Ingress
Incoming Queue (messages)
34554
Contoso Blob Storage Latency
150 ms
100 ms
50 ms
0 ms
20:00
21:00
22:00
contosobackendvmdiag.SuccessE2ELatency
Contoso backend VMSS avg. CPU%
1.25
1.00
0.75
0.50
0.25
20:00
21:00
22:00
demovmss.Percentage CPU
Contoso Workflow requests
1.5
1.0
0.5
0
19:30
20:00
20:30
21:00
21:30
22:00
LoanAppLA.TriggersCompleted
LoanAppLA.TriggersStarted
Outgoing Queue (messages)
233367
Contoso Web Incoming Requests by City (AI)
30
25
20
15
10
5
0
19:30
20:00
20:30
21:00
21:30
22:00
requests/count{client/city=""}
requests/count{client/city="Singapore"}
requests/count{client/city="Boydton"}
Contoso Web Request Duration by City (AI)
8 seconds
6 seconds
4 seconds
2 seconds
0 milliseconds
19:30
20:00
20:30
21:00
21:30
22:00
requests/duration{client/city="Singapore"}
requests/duration{client/city=""}
requests/duration{client/city="Boydton"}
Web Avg. Response Time
8 seconds
6 seconds
4 seconds
2 seconds
0 seconds
20:00
21:00
22:00

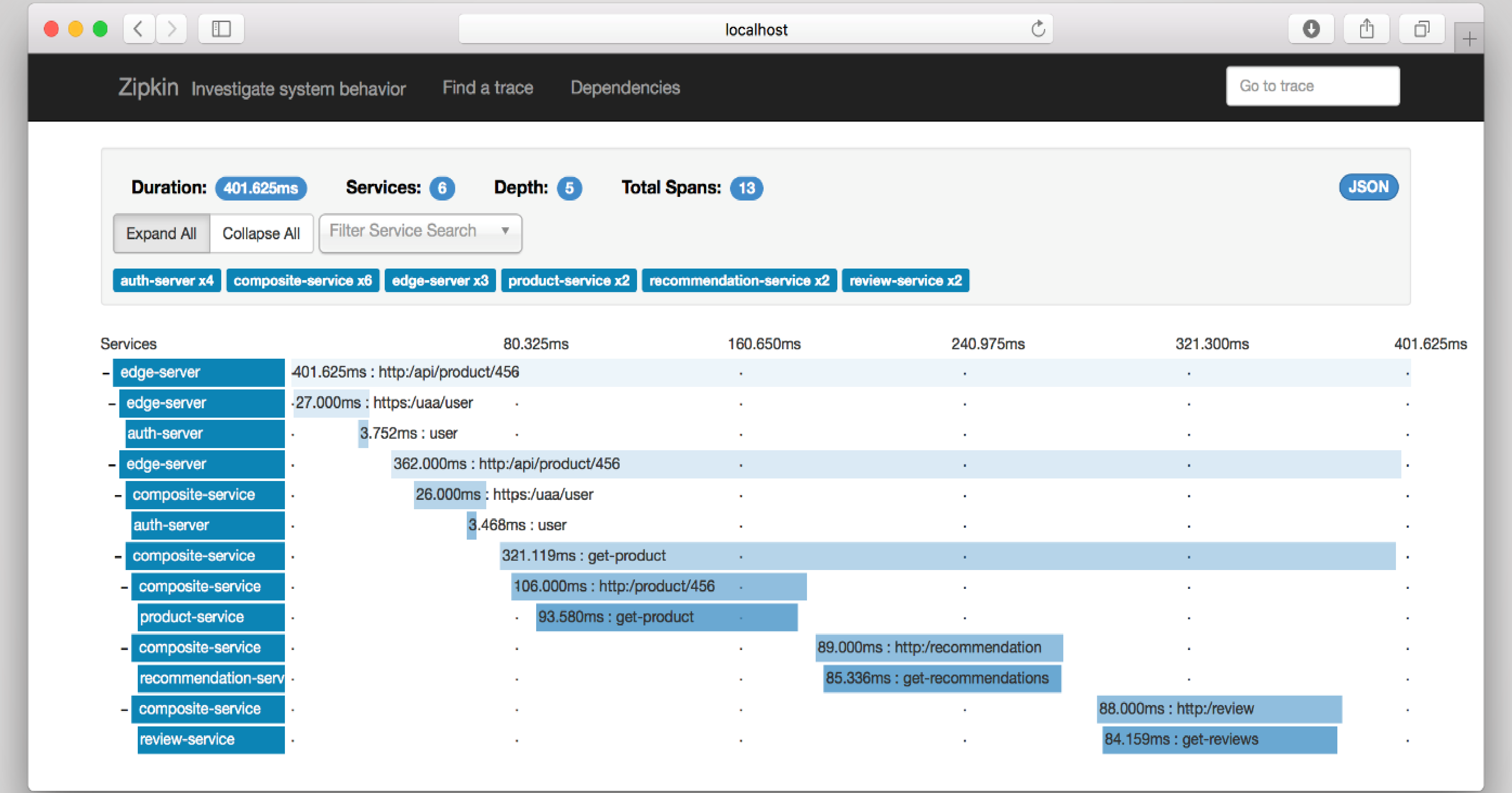
Grafana dashboard



```
October 26th 2018, 17:13:45.000   @timestamp: October 26th 2018, 17:13:45.000 @version: 1 beat.hostname: elastic-stack-18
                                  04 beat.name: elastic-stack-1804 beat.version: 6.4.2 input.type: log host.name: elasti
                                  c-stack-1804 source: /var/log/syslog tags: beats_input_codec_plain_applied offset: 248,
                                  899 prospector.type: log system.syslog.program: kernel system.syslog.hostname: elastic-
                                  stack-1804 system.syslog.message: [ 3907.493183] [UFW BLOCK] IN=eth0 OUT= MAC=56:b0:47:4

October 26th 2018, 17:13:38.000   @timestamp: October 26th 2018, 17:13:38.000 beat.hostname: elastic-stack-1804
                                  beat.name: elastic-stack-1804 beat.version: 6.4.2 @version: 1 input.type: log
                                  host.name: elastic-stack-1804 source: /var/log/syslog tags: beats_input_codec_plain_app
                                  lied offset: 248,631 prospector.type: log system.syslog.program: kernel
```

Kibana dashboard



Duration: 401.625ms Services: 6 Depth: 5 Total Spans: 13

auth-server x4 composite-service x6 edge-server x3 product-service x2 recommendation-service x2 review-service x2

Services	80.325ms	160.650ms	240.975ms	321.300ms	401.625ms
– edge-server	401.625ms : http:/api/product/456				
– edge-server	27.000ms : https:/uaa/user				
auth-server	3.752ms : user				
– edge-server	362.000ms : http:/api/product/456				
– composite-service	26.000ms : https:/uaa/user				
auth-server	3.468ms : user				
– composite-service	321.119ms : get-product				
– composite-service	106.000ms : http:/product/456				
product-service	93.580ms : get-product				
– composite-service			89.000ms : http:/recommendation		
recommendation-serv			85.336ms : get-recommendations		
– composite-service				88.000ms : http:/review	
review-service				84.159ms : get-reviews	

Zipkin tracing tool

Portability checklist

Must have:

- Enable containerization
- Use immutable tags
- Follow to best practices for images
- Use external configuration
- Use versioning
- Don't embed infrastructure into services

Should have:

- Define quotas for CPU and memory

Security checklist

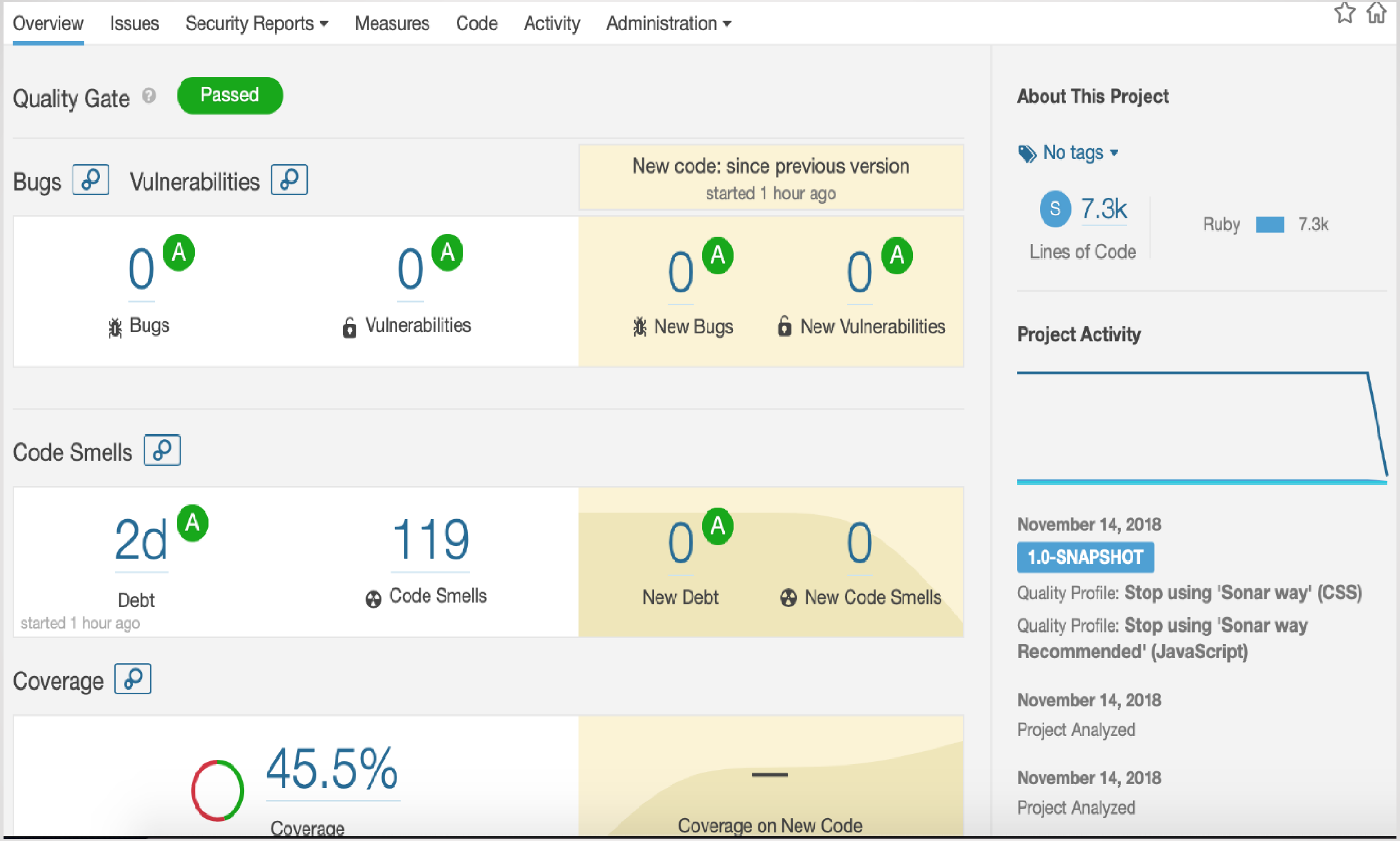
Must have:

- Segregate services by security traits
- Validate inbound data
- Don't expose sensitive data
- Control dependencies versions

Maintainability checklist

Must have:

- Use branching strategy
- Enable build automation
- Use unit tests
- Define feedback activities
- Use code conventions
- Reduce code duplication
- Remove dead code
- Ensure methods and classes maintainability



Overview Issues Security Reports ▾ Measures Code Activity Administration ▾

Quality Gate Passed

Bugs Vulnerabilities

New code: since previous version
started 1 hour ago

0 A Bugs

0 A Vulnerabilities

0 A New Bugs

0 A New Vulnerabilities

Code Smells

2d A Debt
started 1 hour ago

119 Code Smells

0 A New Debt

0 New Code Smells

Coverage

45.5% Coverage

— Coverage on New Code

About This Project

No tags ▾

S 7.3k Lines of Code

Ruby 7.3k

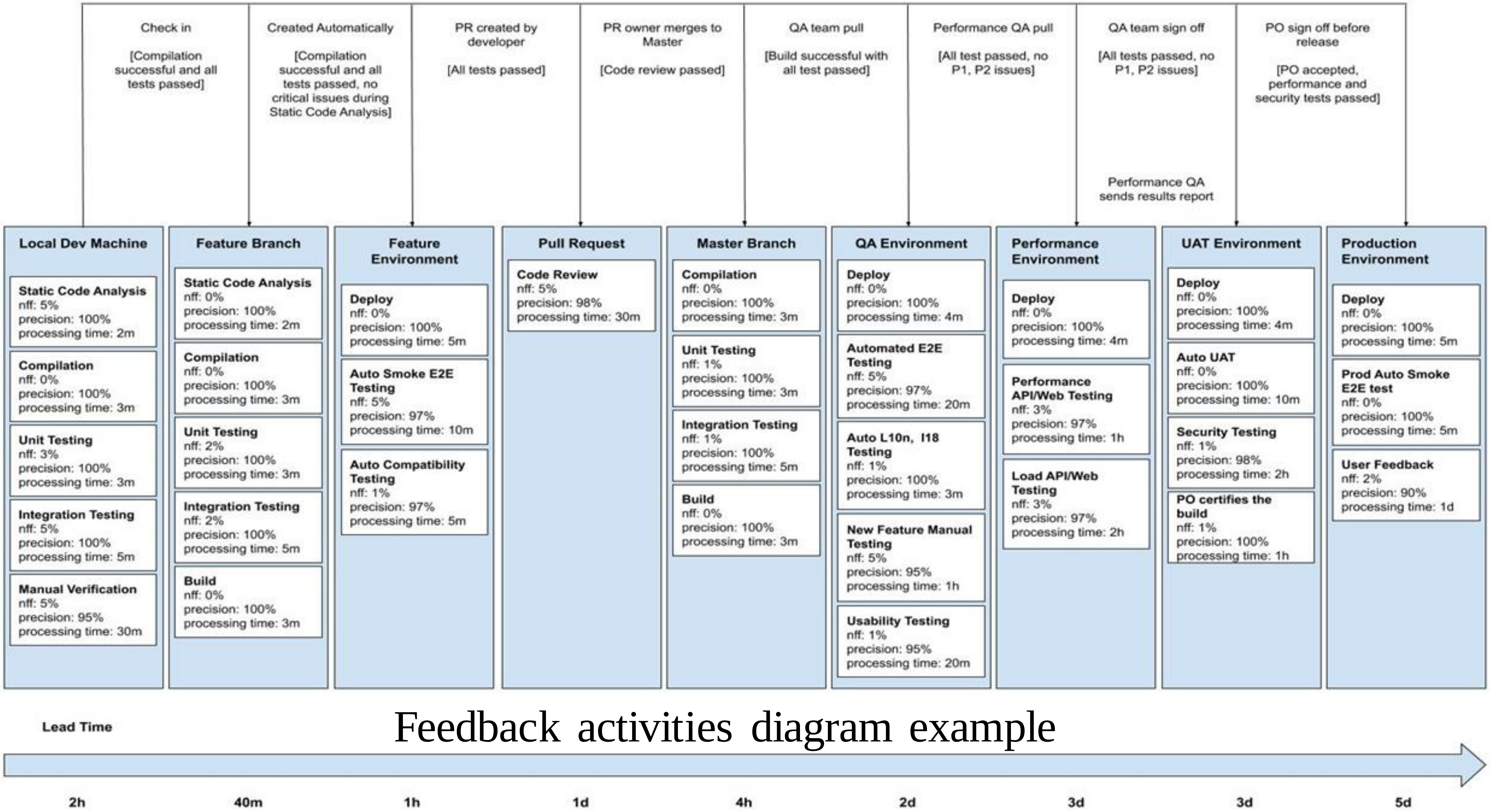
Project Activity

November 14, 2018
1.0-SNAPSHOT
Quality Profile: Stop using 'Sonar way' (CSS)
Quality Profile: Stop using 'Sonar way Recommended' (JavaScript)

November 14, 2018
Project Analyzed

November 14, 2018
Project Analyzed

SonarQube dashboard



Check in

[Compilation successful and all tests passed]

Created Automatically

[Compilation successful and all tests passed, no critical issues during Static Code Analysis]

PR created by developer

[All tests passed]

PR owner merges to Master

[Code review passed]

QA team pull

[Build successful with all test passed]

Performance QA pull

[All test passed, no P1, P2 issues]

QA team sign off

[All tests passed, no P1, P2 issues]

Performance QA sends results report

PO sign off before release

[PO accepted, performance and security tests passed]

Local Dev Machine

- **Static Code Analysis** nff: 5% precision: 100% processing time: 2m
- **Compilation** nff: 0% precision: 100% processing time: 3m
- **Unit Testing** nff: 3% precision: 100% processing time: 3m
- **Integration Testing** nff: 5% precision: 100% processing time: 5m
- **Manual Verification** nff: 5% precision: 95% processing time: 30m

Feature Branch

- **Static Code Analysis** nff: 0% precision: 100% processing time: 2m
- **Compilation** nff: 0% precision: 100% processing time: 3m
- **Unit Testing** nff: 2% precision: 100% processing time: 3m
- **Integration Testing** nff: 2% precision: 100% processing time: 5m
- **Build** nff: 0% precision: 100% processing time: 3m

Feature Environment

- **Deploy** nff: 0% precision: 100% processing time: 5m
- **Auto Smoke E2E Testing** nff: 5% precision: 97% processing time: 10m
- **Auto Compatibility Testing** nff: 1% precision: 97% processing time: 5m

Pull Request

- **Code Review** nff: 5% precision: 98% processing time: 30m

Master Branch

- **Compilation** nff: 0% precision: 100% processing time: 3m
- **Unit Testing** nff: 1% precision: 100% processing time: 3m
- **Integration Testing** nff: 1% precision: 100% processing time: 5m
- **Build** nff: 0% precision: 100% processing time: 3m

QA Environment

- **Deploy** nff: 0% precision: 100% processing time: 4m
- **Automated E2E Testing** nff: 5% precision: 97% processing time: 20m
- **Auto L10n, I18 Testing** nff: 1% precision: 100% processing time: 3m
- **New Feature Manual Testing** nff: 5% precision: 95% processing time: 1h
- **Usability Testing** nff: 1% precision: 95% processing time: 20m

Performance Environment

- **Deploy** nff: 0% precision: 100% processing time: 4m
- **Performance API/Web Testing** nff: 3% precision: 97% processing time: 1h
- **Load API/Web Testing** nff: 3% precision: 97% processing time: 2h

UAT Environment

- **Deploy** nff: 0% precision: 100% processing time: 4m
- **Auto UAT** nff: 0% precision: 100% processing time: 10m
- **Security Testing** nff: 1% precision: 98% processing time: 2h
- **PO certifies the build** nff: 1% precision: 100% processing time: 1h

Production Environment

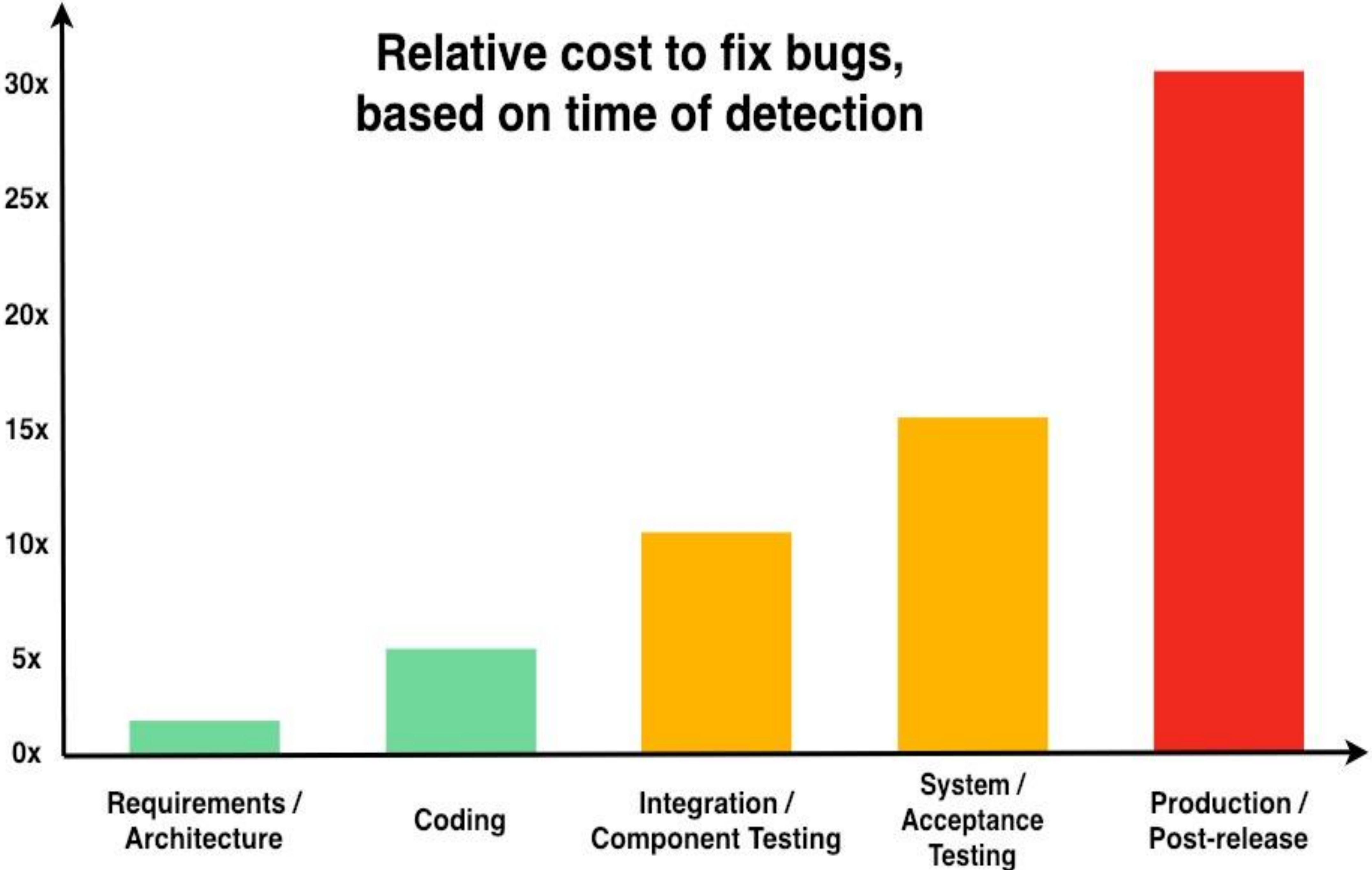
- **Deploy** nff: 0% precision: 100% processing time: 5m
- **Prod Auto Smoke E2E test** nff: 0% precision: 100% processing time: 5m
- **User Feedback** nff: 2% precision: 90% processing time: 1d

Lead Time

2h 40m 1h 1d 4h 2d 3d 3d 5d

Feedback activities diagram example

Relative cost to fix bugs, based on time of detection



Well defined criteria will help to conduct audit smoothly and find out most of technical dept.
It does not guaranty success of product or absence of problems but properly conducted technical audit reduce risk to have them after release