Cloud Native Migration in a Sales-driven Department

@JUG Karlsruhe - 2018-09-12

Marcel in a nutshell

- Marcel Becker
- Platform Developer
  @ 1&1 Mail&Media Development & Technology GmbH
- Spring, Cloud & CI/CD
- https://github.com/tuxdevelop/

Simon in a nutshell

- Simon Frettlöh
- Software Architect
  @ 1&1 Mail&Media Development & Technology GmbH
- External Lecturer
  @ Duale Hochschule Baden-Württemberg (DHBW)

Agenda

- The organisation
- The status quo in summer 2017
- The vision
- What we actually did
- Where we want to go

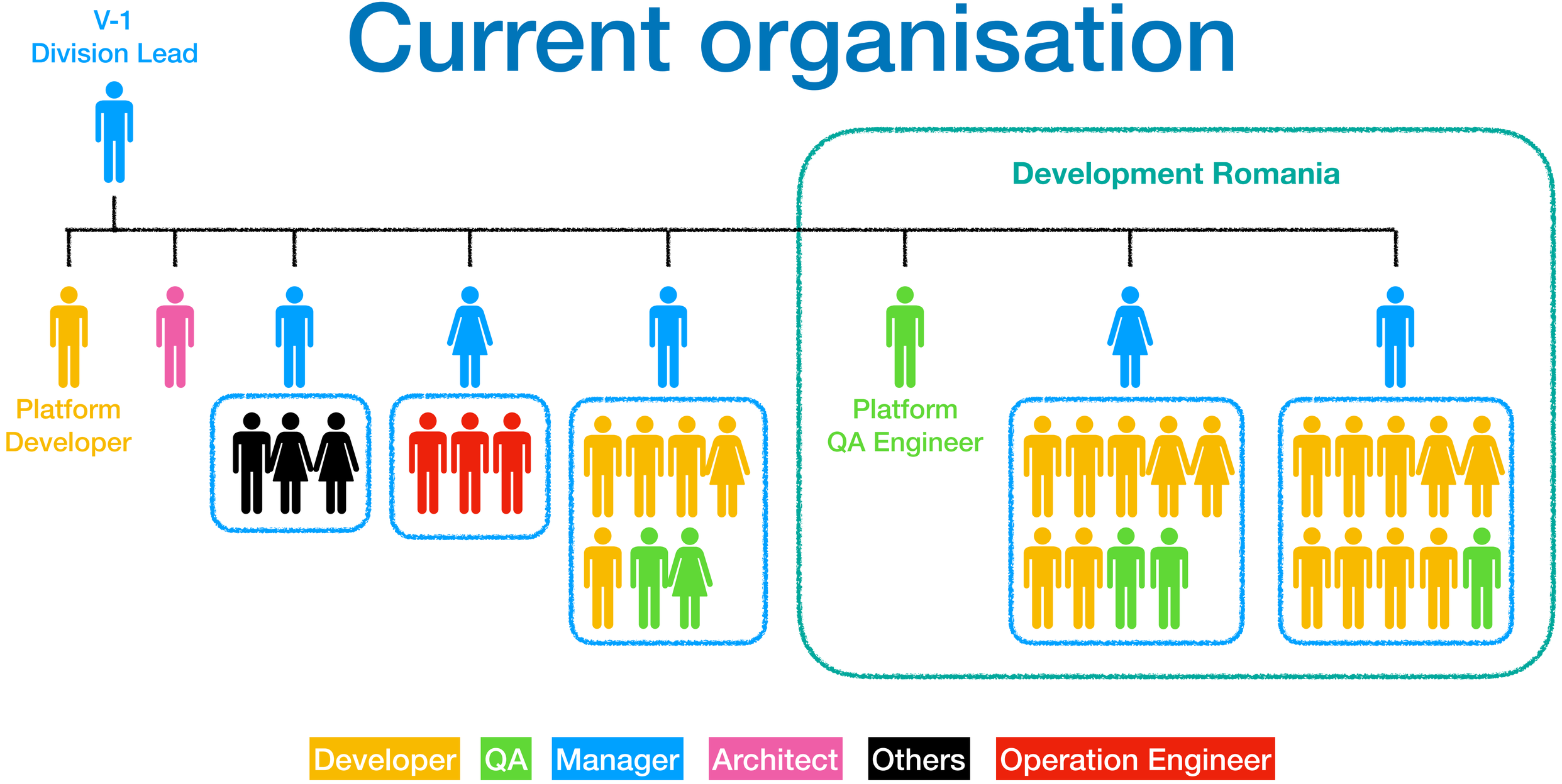
The organisation

Where did we come from?

Anno 2011

Developer | Test Automation Engineer | Architect | Others | Operation Engineer

Current organisation



Business grows, too

- Partner Products
- Energy (Electricity & Gas)
- Mobile Phone Shop (web.de & GMX)
- Pay-Mail Products
  (web.de, GMX, mail.com)
- WEB.Cent Platform
- Lead Platform
- Customer Care Systems
- De-Mail Business Processes
- Content Services

42 people from 6 Departments

own management board

22 DEVs

6 QAs

3 OPs

The organisational challenges

22 DEVs

6 QAs

3 OPs

- Bi-weekly prioritisation of more than 3 virtual teams
- ~40 business tasks
- ~30 deployments

Technical Roadmap?
Technical Debts?

The status quo in summer 2017

Outlined architecture

Load Balancer

50 hosts per environment

40 deployables

Shops / Frontends

CMS

Campaign & Product Configuration

Facades / Gateways

Brokers

Storage Backends

Scheduling / Batches

Process Platforms

~80 External Services

Technologies …

4.2.3.GA

Load Balancer

50 hosts per environment

hazelcast

Shops / Frontends

CMS

HIPPO

Campaign & Product Configuration

Facades / Gateways

Brokers

Storage Backends

Scheduling / Batches

Process Platforms

… and versions

Spring Core
- 3.2.x
- 4.1.x
- 4.2.x
- 4.3.x

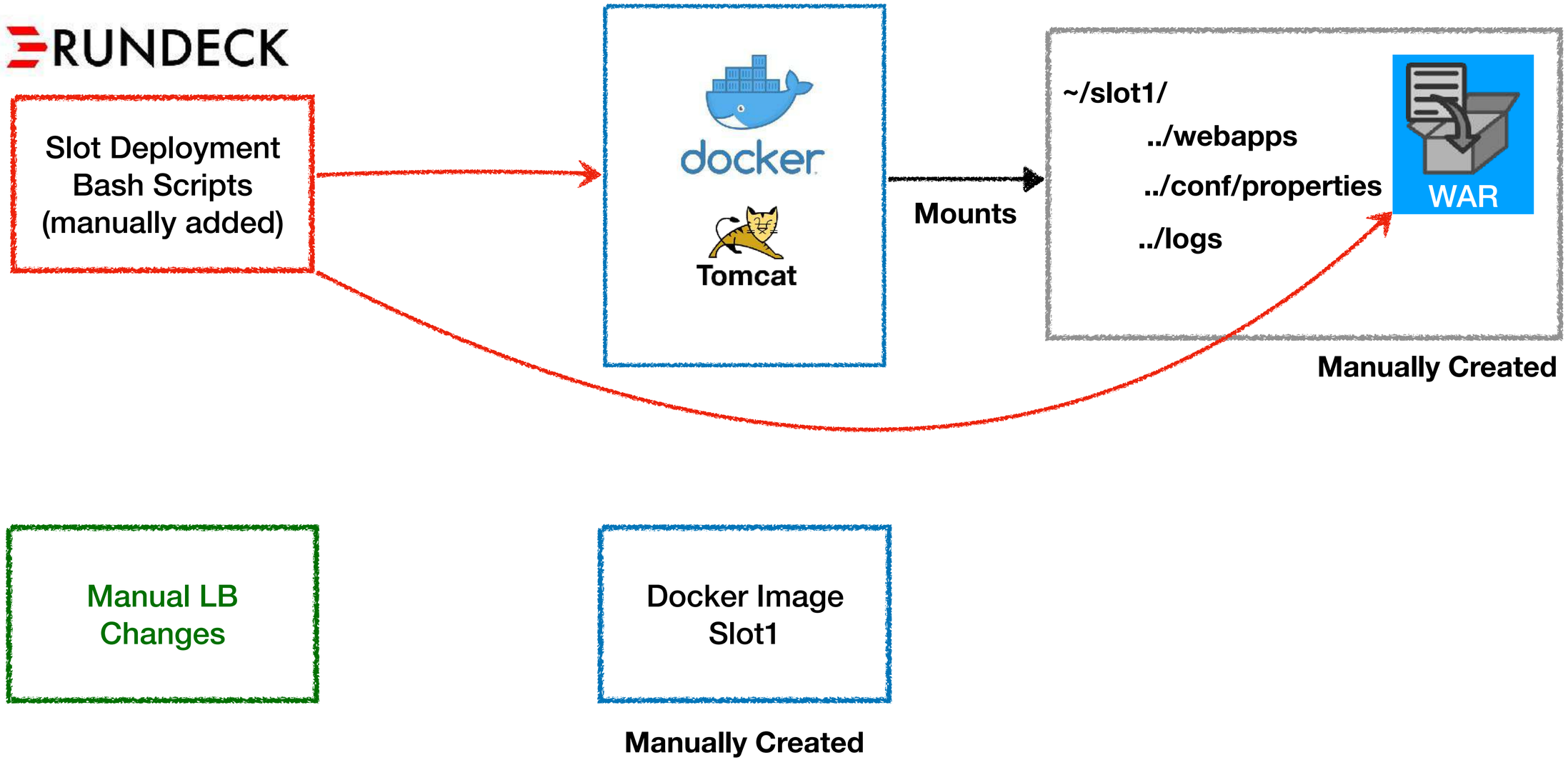
Spring Boot
- 1.1.x
- 1.2.x
- 1.4.x

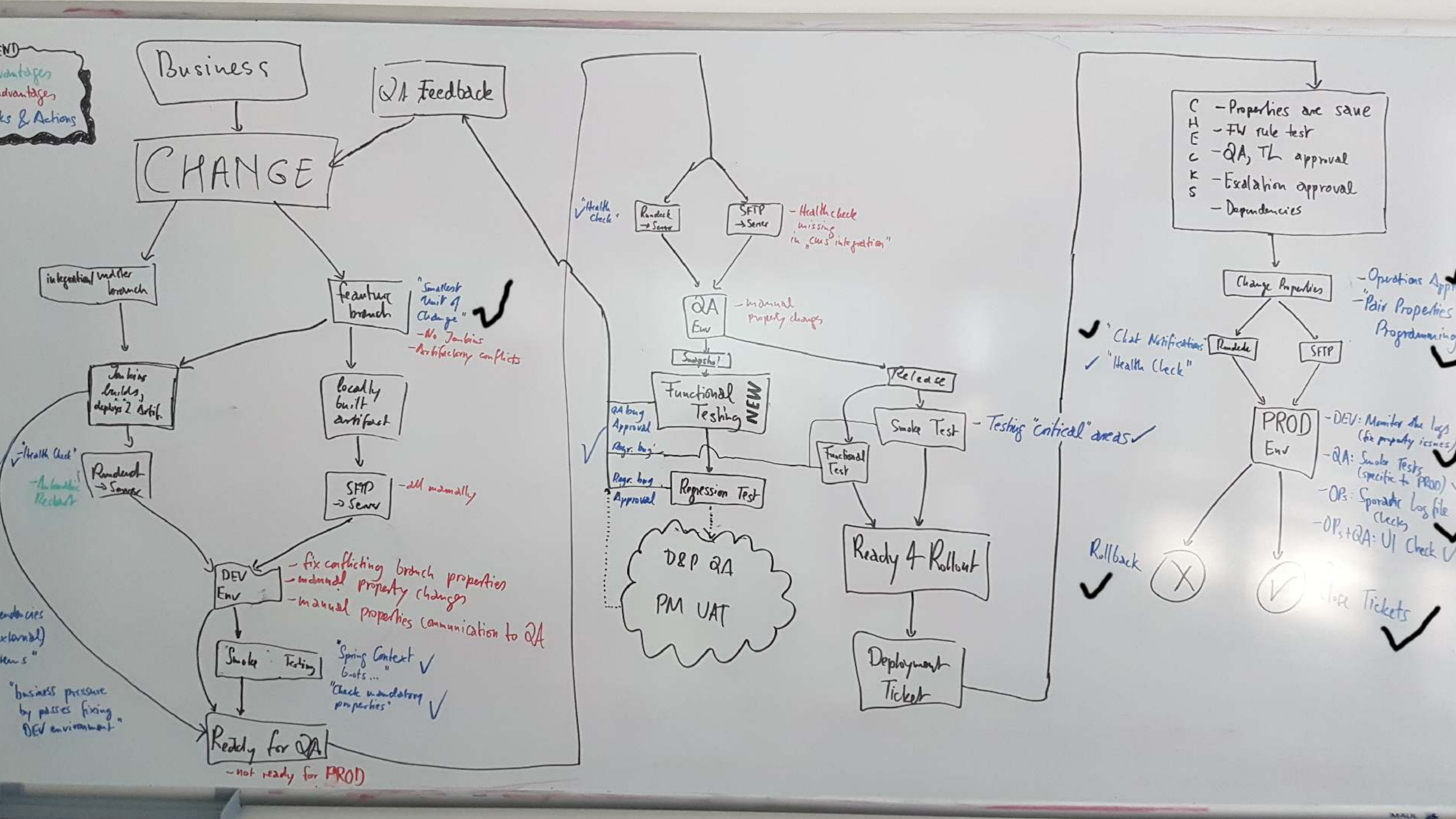
Activiti 5.11
released 2012

No unified stack

No common upgrade strategy

Partially automated deployment




Business
QA Feedback
CHANGE
integration/master branch
feature branch
"Smallest Unit of Change"
- No Jenkins
- Artifactory conflicts
Jenkins builds, deploys 2 Artif.
locally built artifact
"Health Check"
Rundeck → Server
SFTP → Server
- all manually
DEV Env
- fix conflicting branch properties
- manual property changes
- manual properties communication to QA
Smoke Testing
"Spring Context boots..."
"Check mandatory properties"
Ready for QA
- not ready for PROD
"business pressure by passes fixing DEV environment"
"Health Check"
Rundeck → Server
SFTP → Server
- Health checks missing in „CMS integration"
QA Env
- manual property changes
Snapshot
Functional Testing NEW
QA bug Approval
Regr. bug
Regr. bug Approval
Regression Test
D&P QA
PM UAT
Release
Smoke Test
- Testing "critical" areas
Functional Test
Ready 4 Rollout
Deployment Ticket
CHECKS
- Properties are save
- FW rule test
- QA, TL approval
- Escalation approval
- Dependencies
Change Properties
- Operations Appr
- Pair Properties Programming
"Chat Notifications"
"Health Check"
Rundeck
SFTP
PROD Env
- DEV: Monitor the logs (fix property issues)
- QA: Smoke Tests (specific to PROD)
- OPs: Sporadic Log file checks
- OPs+QA: UI Check
Rollback
Tickets

Technical debt & challenges

No 100% deployment unification

- property files different across nodes
- DEV, QA and PROD differ
- hard to provide new environments

Large effort for adding new applications

- many features were added to existing applications
- OPs surprised by new features
- software maintenance effort grew
- slower feature rollouts

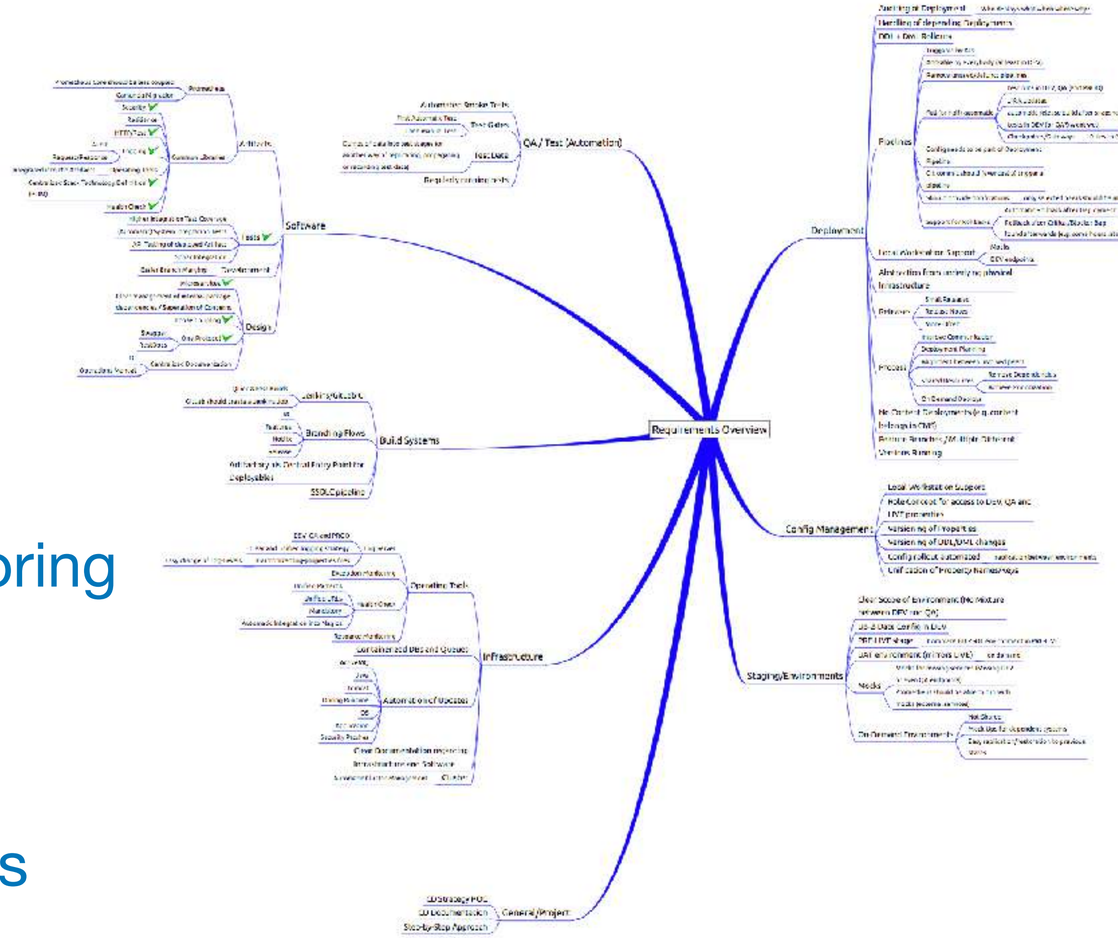
Intransparent infrastructure

- infrastructure components outdated
- implicit expectations, e.g. FW rules

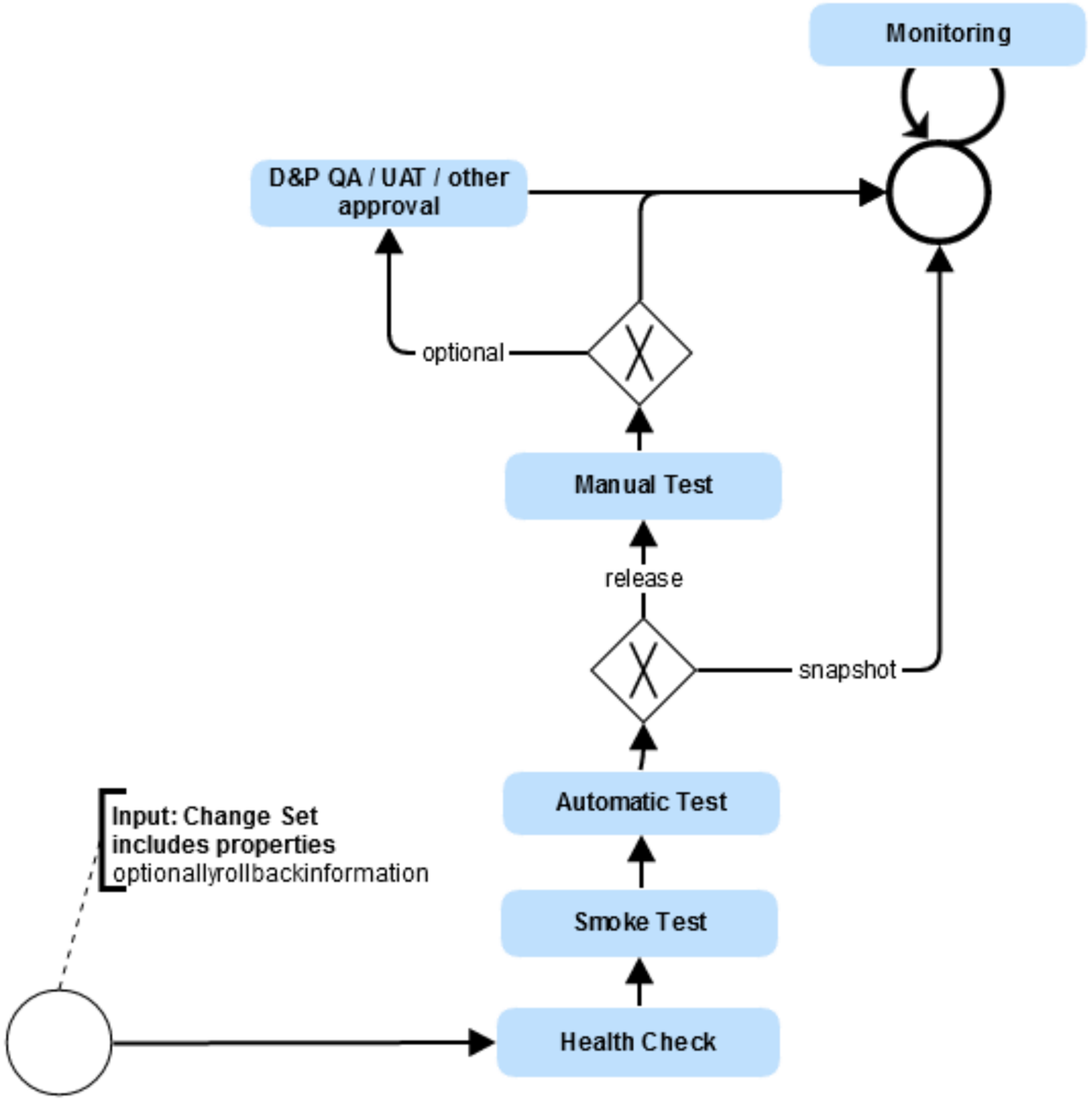
The vision

Expectations & requirements

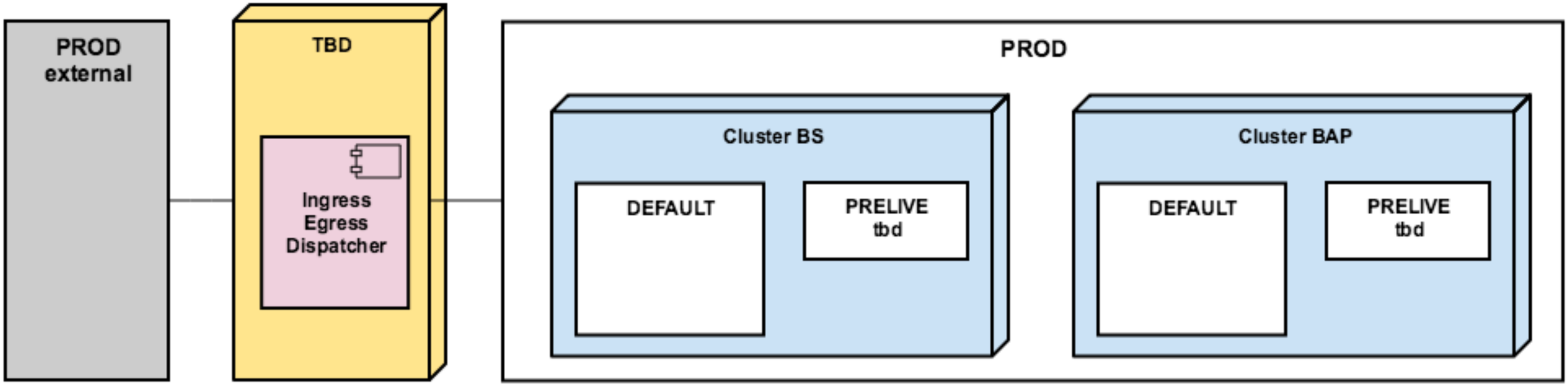
- Automation á la “full DevOps”, including
  - QA automation
  - Configuration as deployable
- Health checks, log server, exception monitoring
- Long-term: hourly deployments
- On-demand test/development environments
- Strong focus on providing business features quickly!



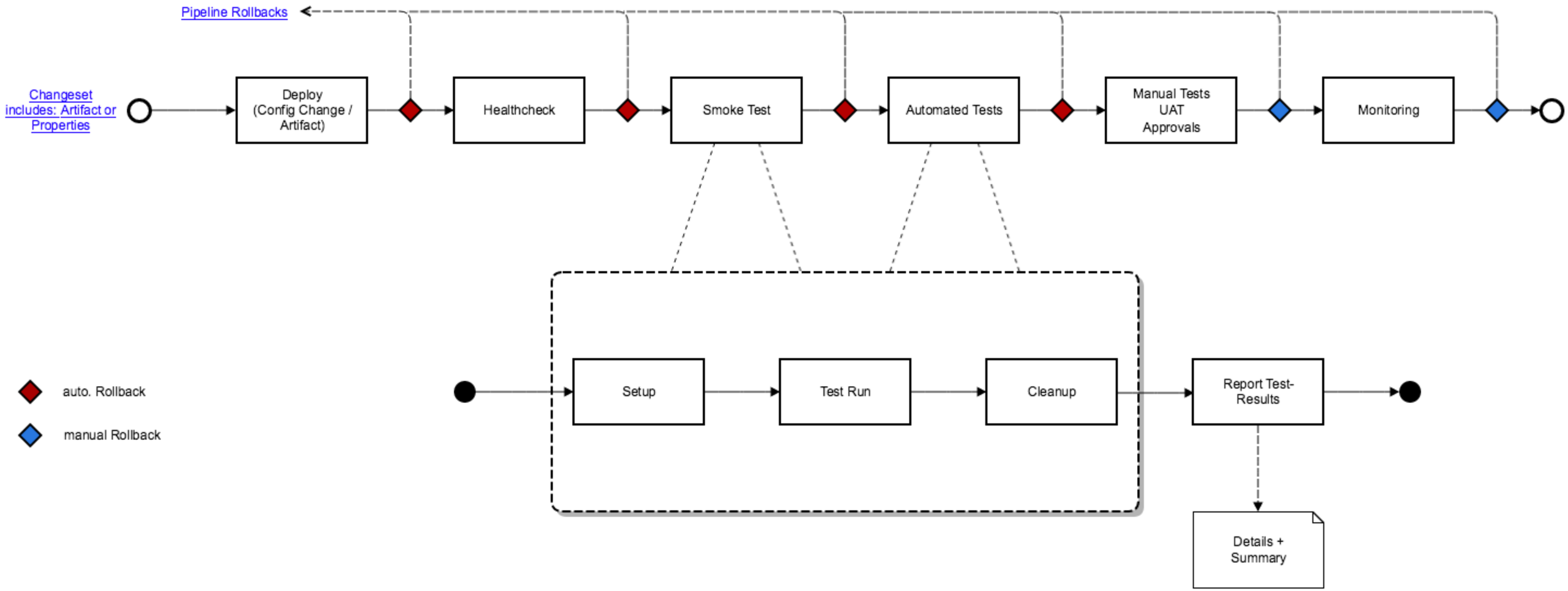
Deployment workflow



Cluster setup



Deployment pipeline



Culture & knowledge

- **Culture**:
  - Shift from regular to DevOps culture
  - Vision: developers can deploy in PROD
  - Stand-by duty & responsibilities
- **Knowledge**:
  - QAs & OPs need to learn more about coding
  - DEVs need to learn more about infrastructure
  - Everybody needs to learn about “cloud native”

But, how did we get stuck?

- Applications far away from being cloud-ready
- DevOps mind-set only amongst few people
- A lot of knowledge was missing
- Time & resources
- “the show must go on”

What we actually did

Application build

Gitlab-CI

Gitlab-ci template Repository

.gitlab-ci.yml

Project

.gitlab-ci.yml

Department Parent POM

Maven Version Policy

Dependency management

Department BOM

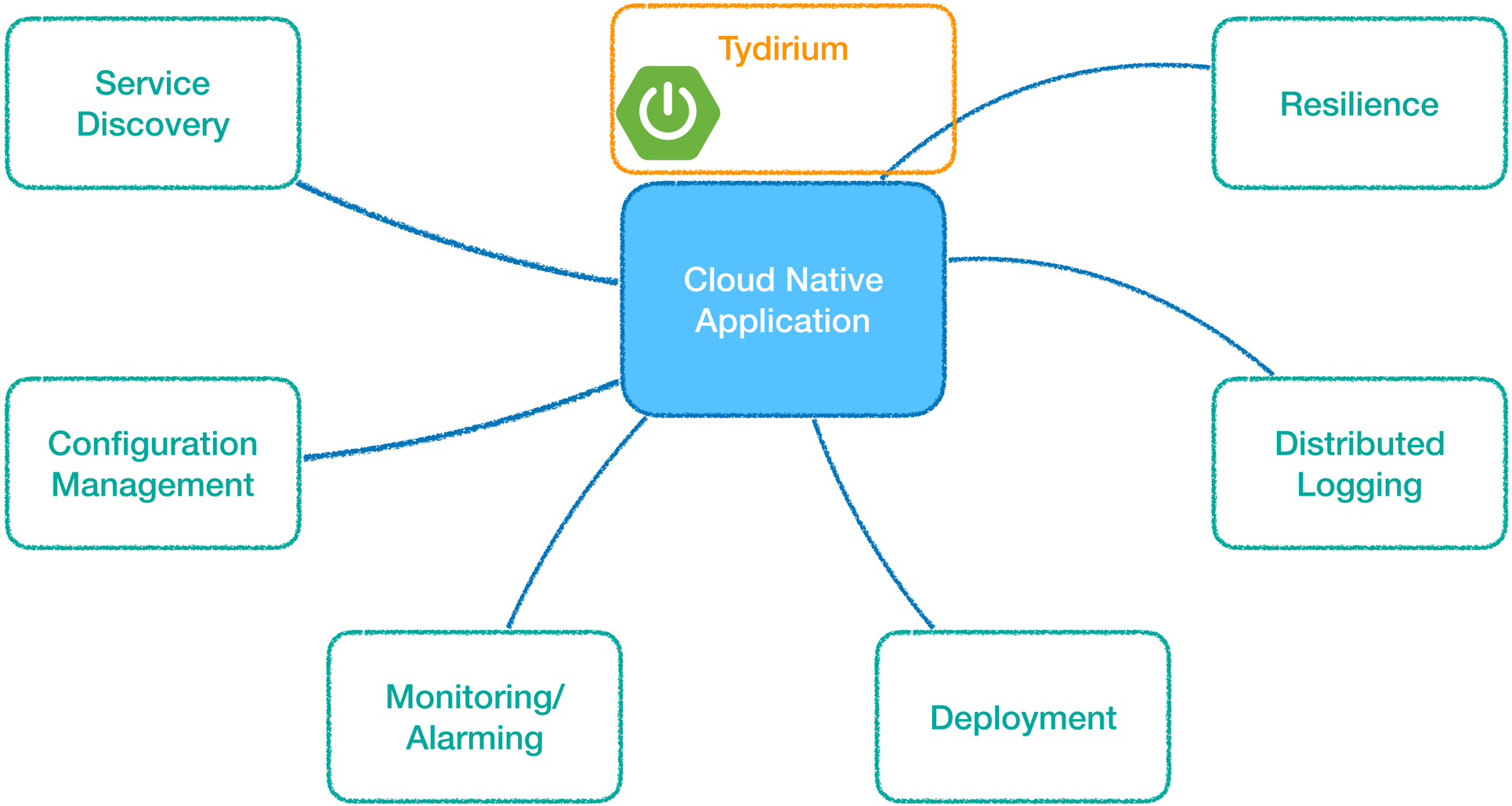
spring.io platform BOM

Spring Cloud Dependencies
Release Train

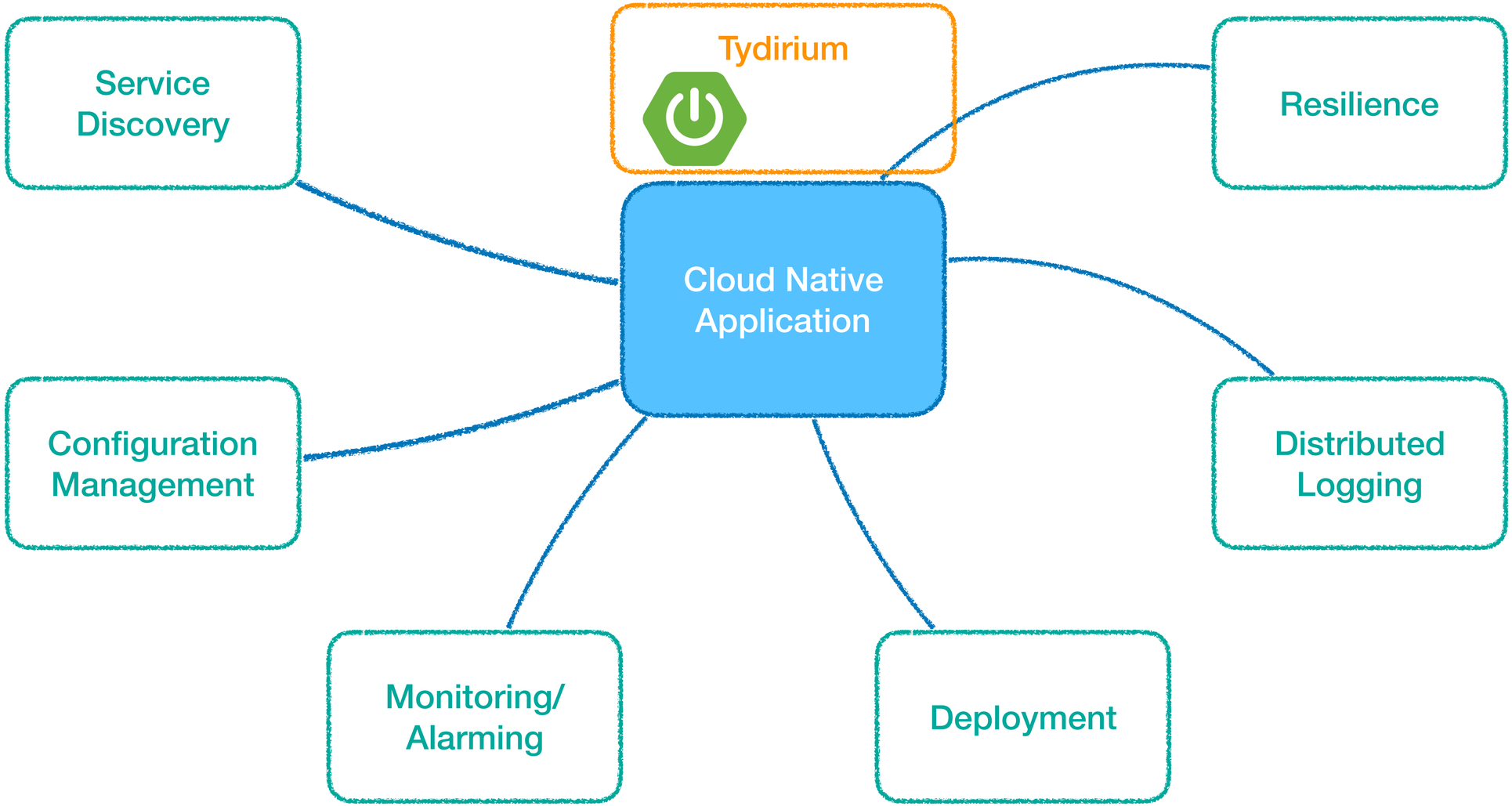
Other Dependencies

~120 dependencies

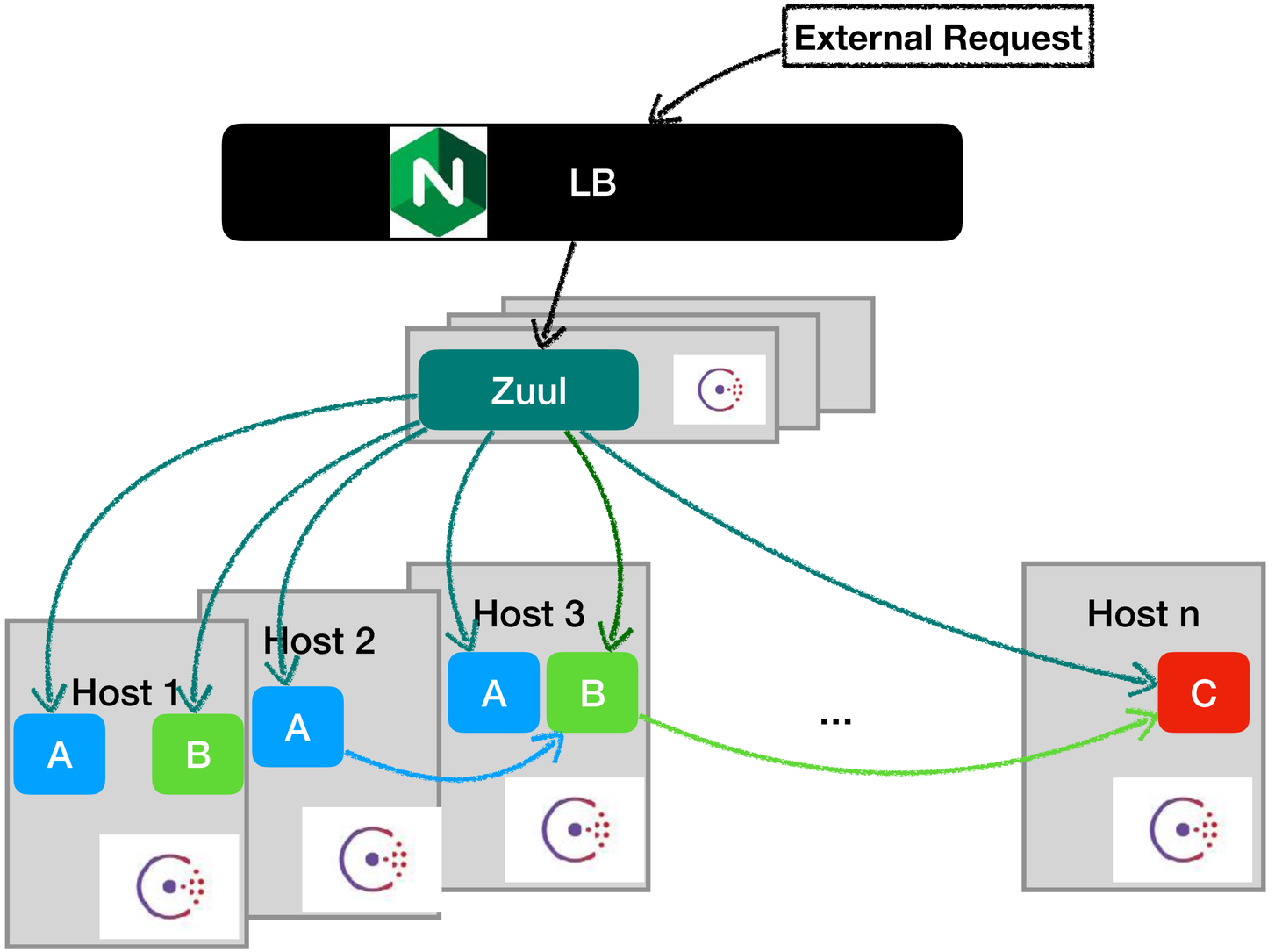
Cloud native stack



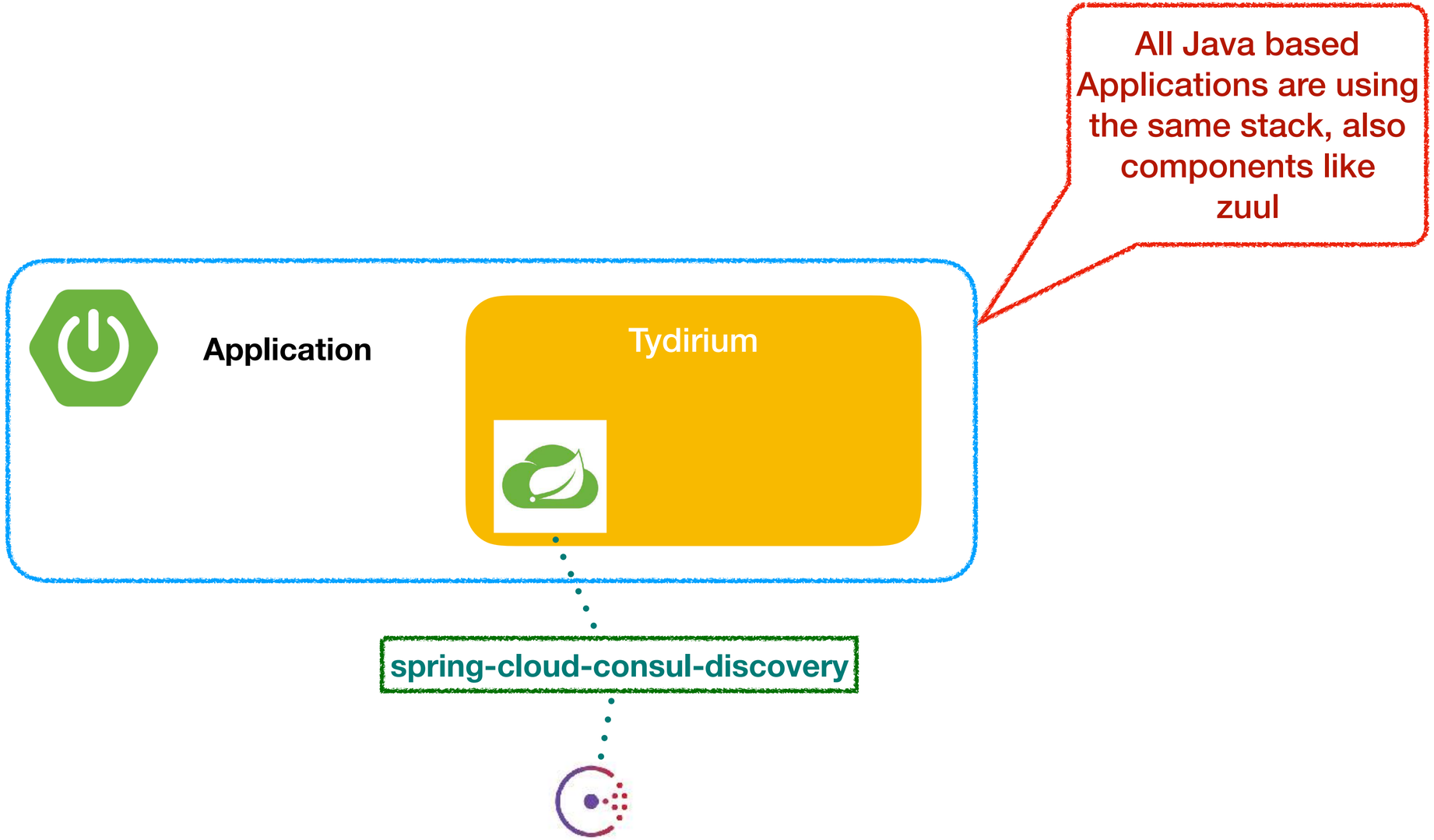
Cloud native stack



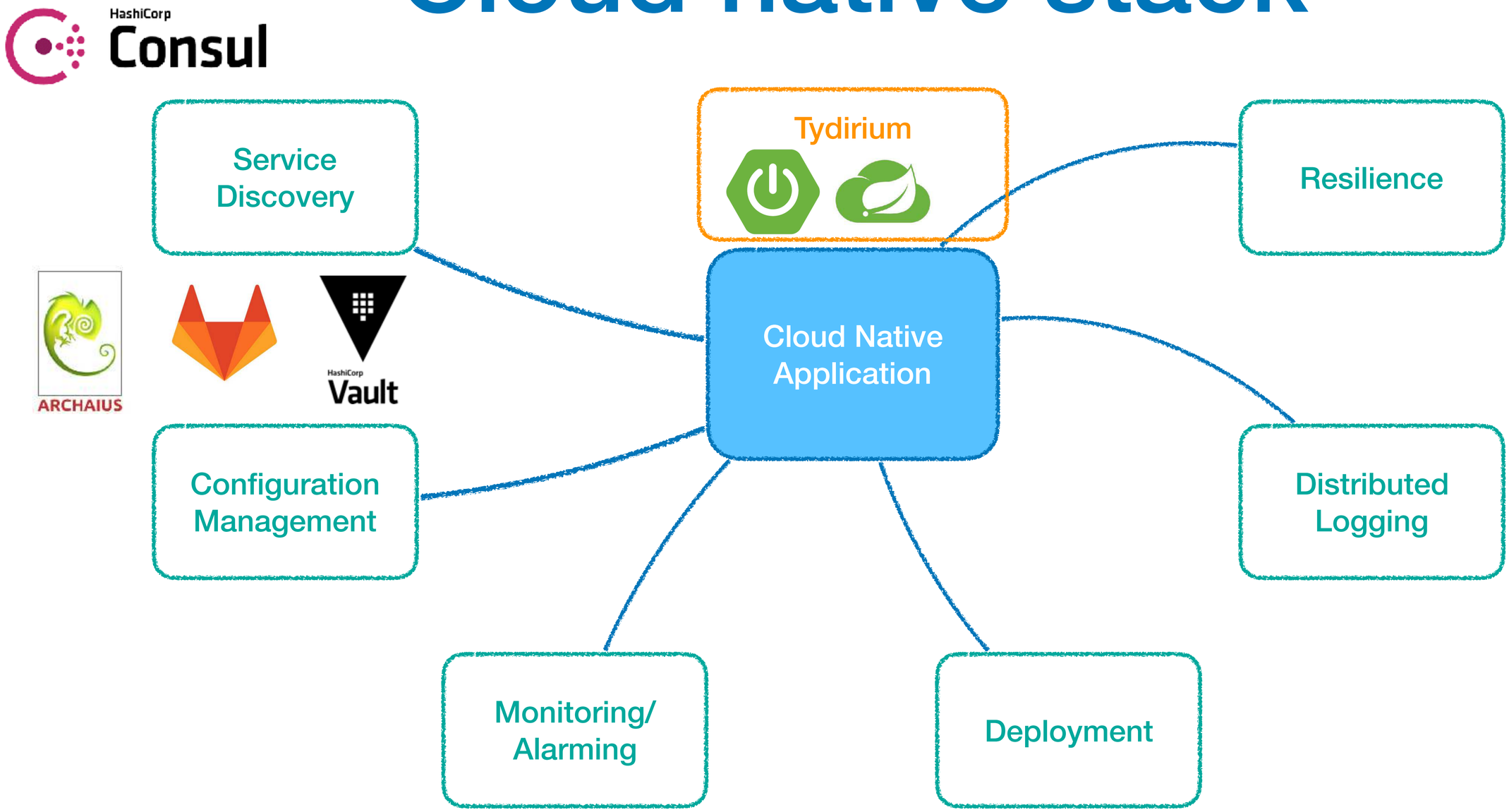
Service discovery



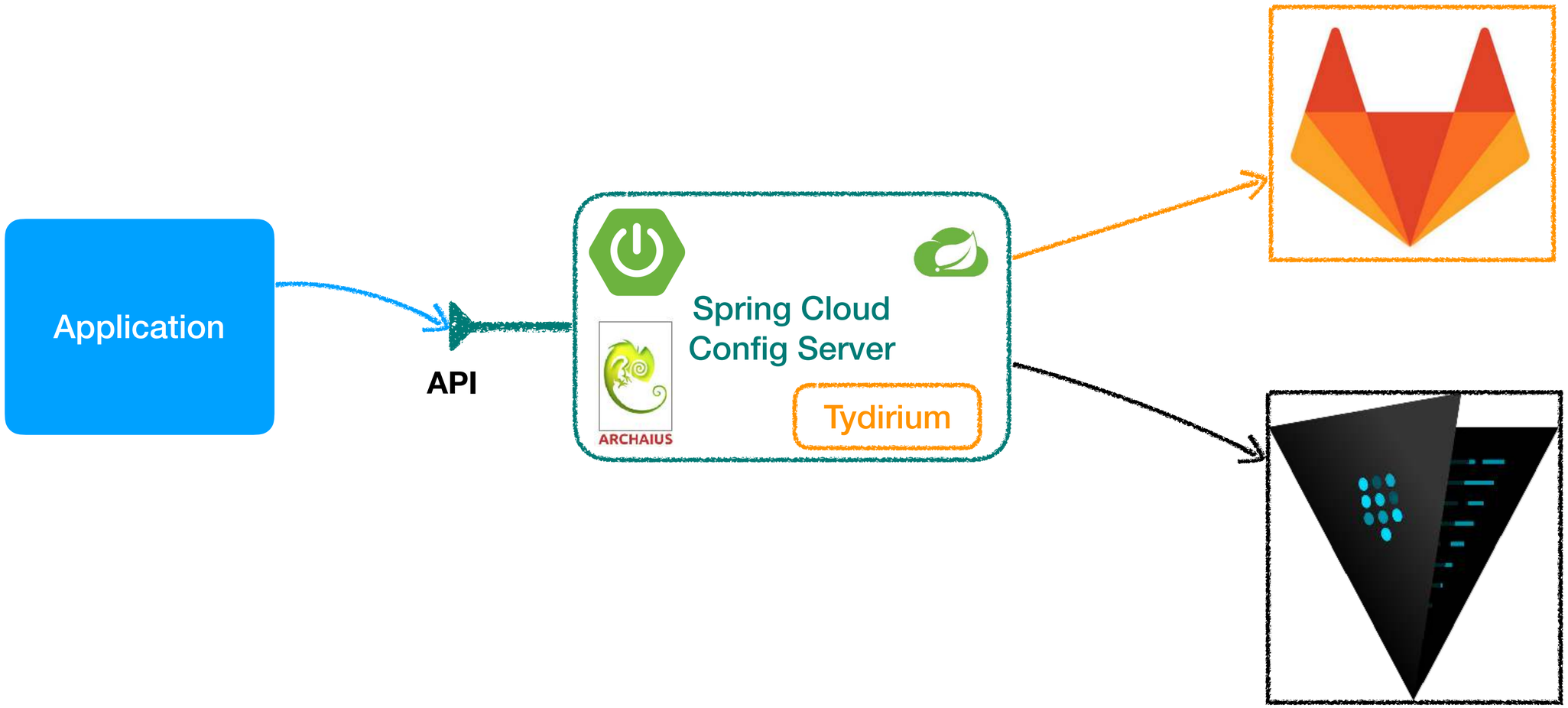
Service discovery



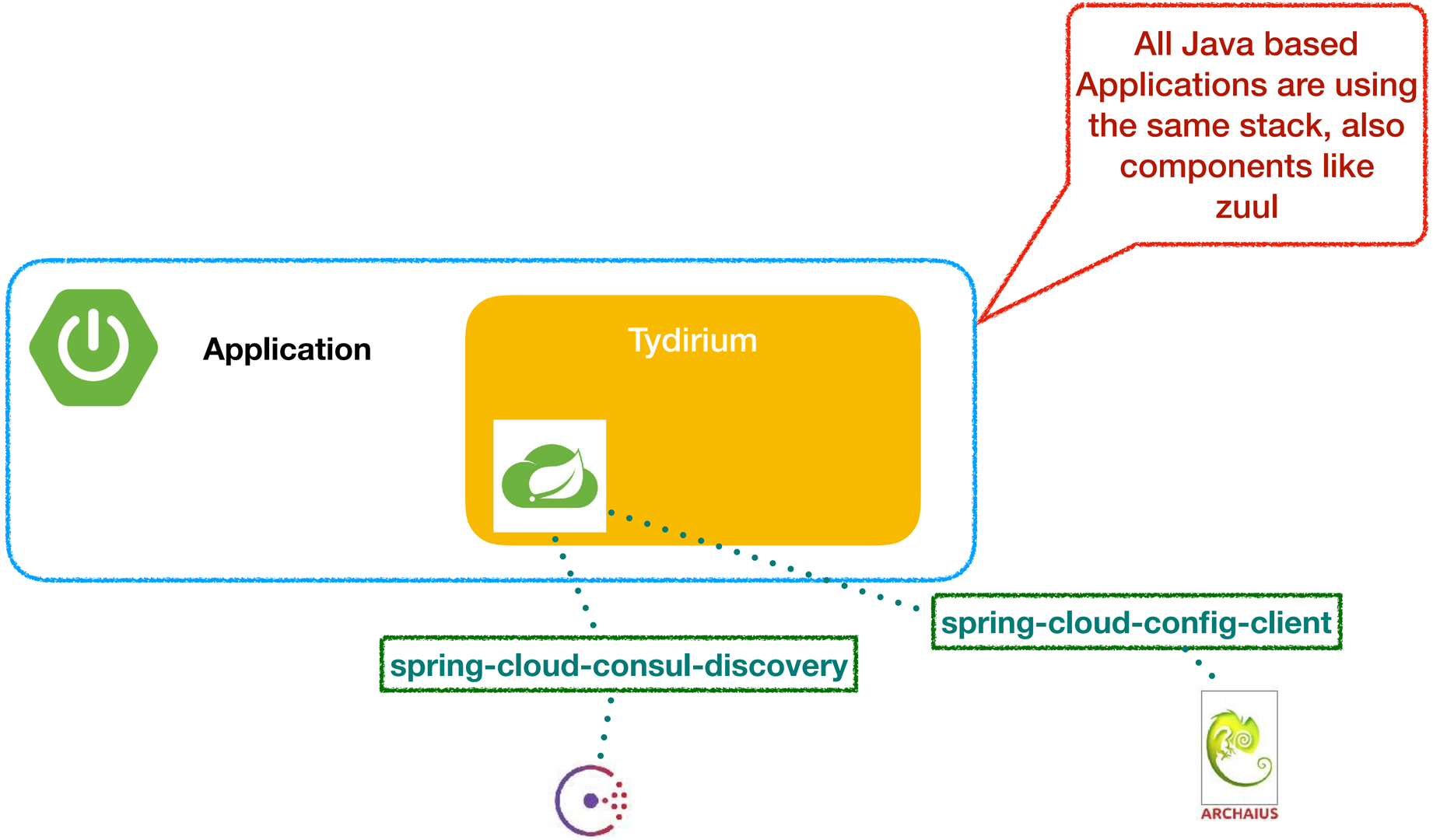
Cloud native stack



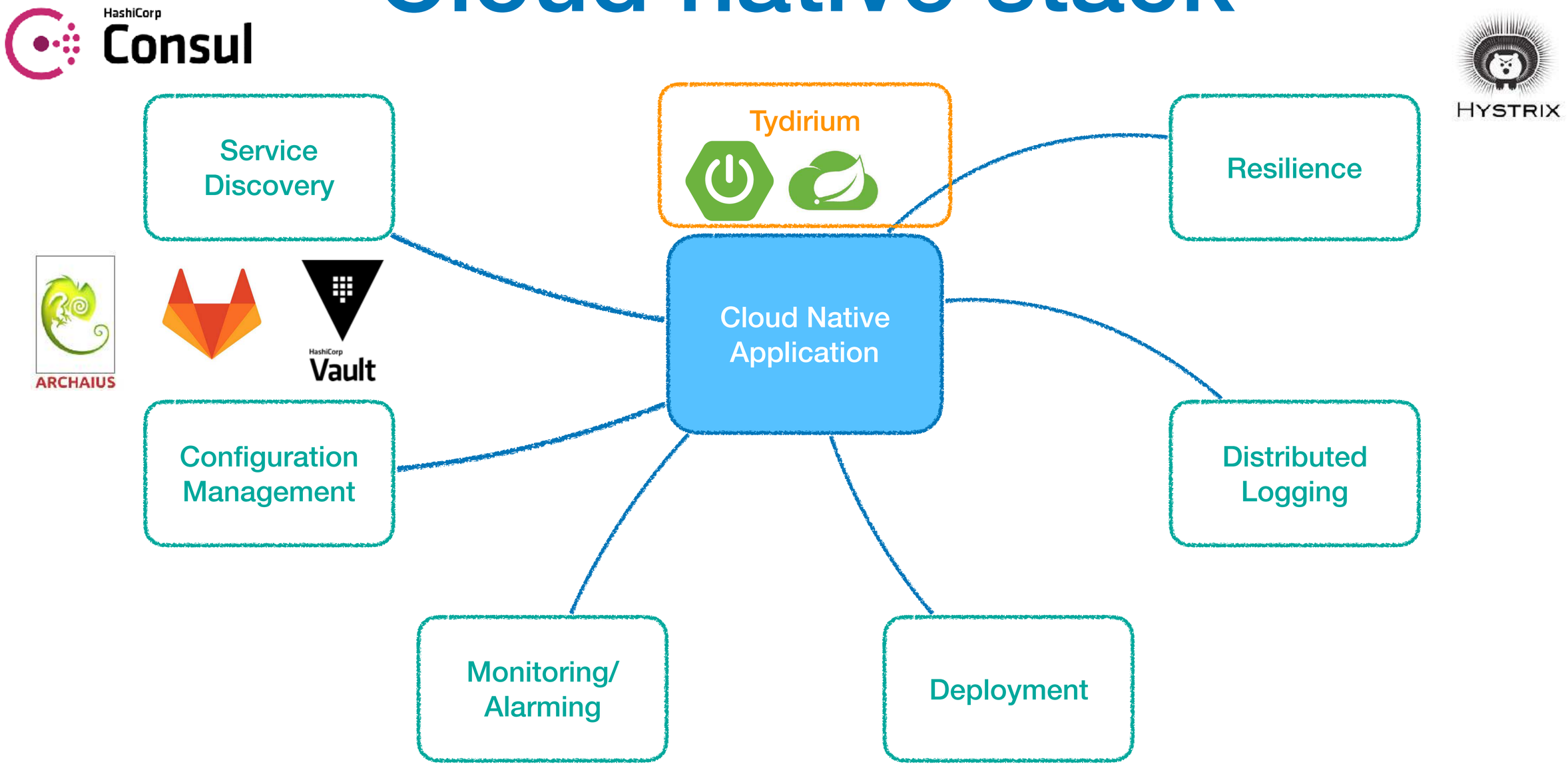
Configuration management



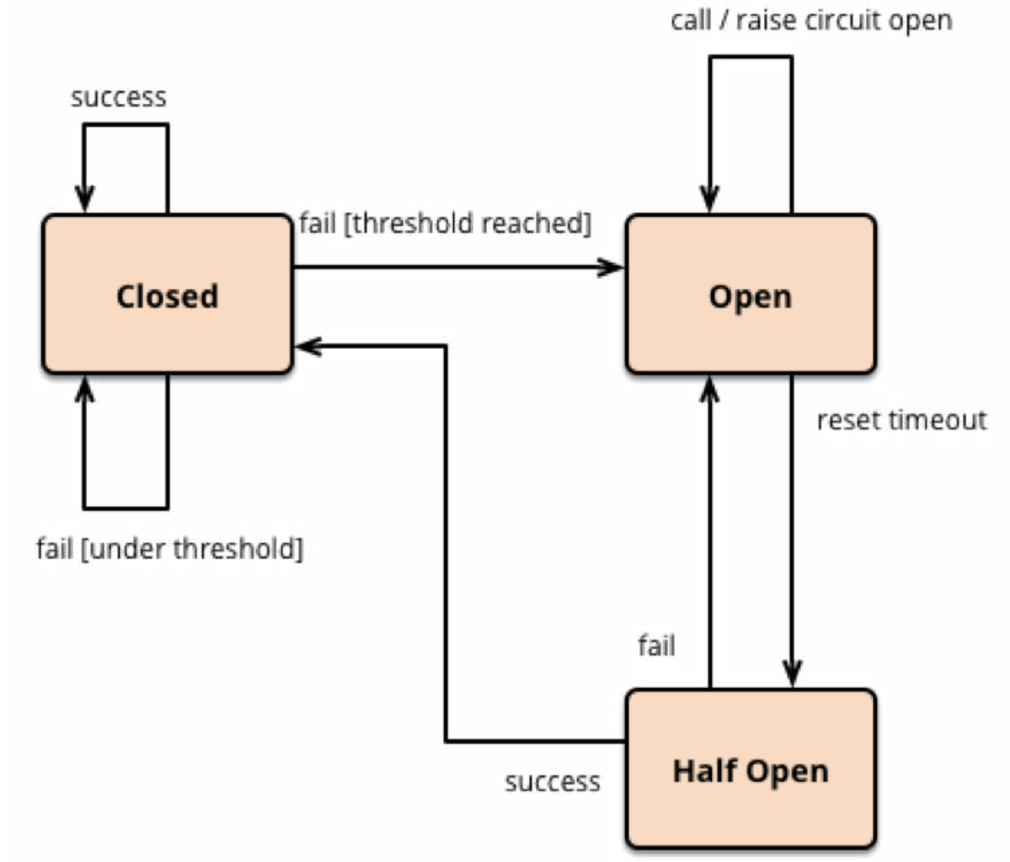
Configuration management

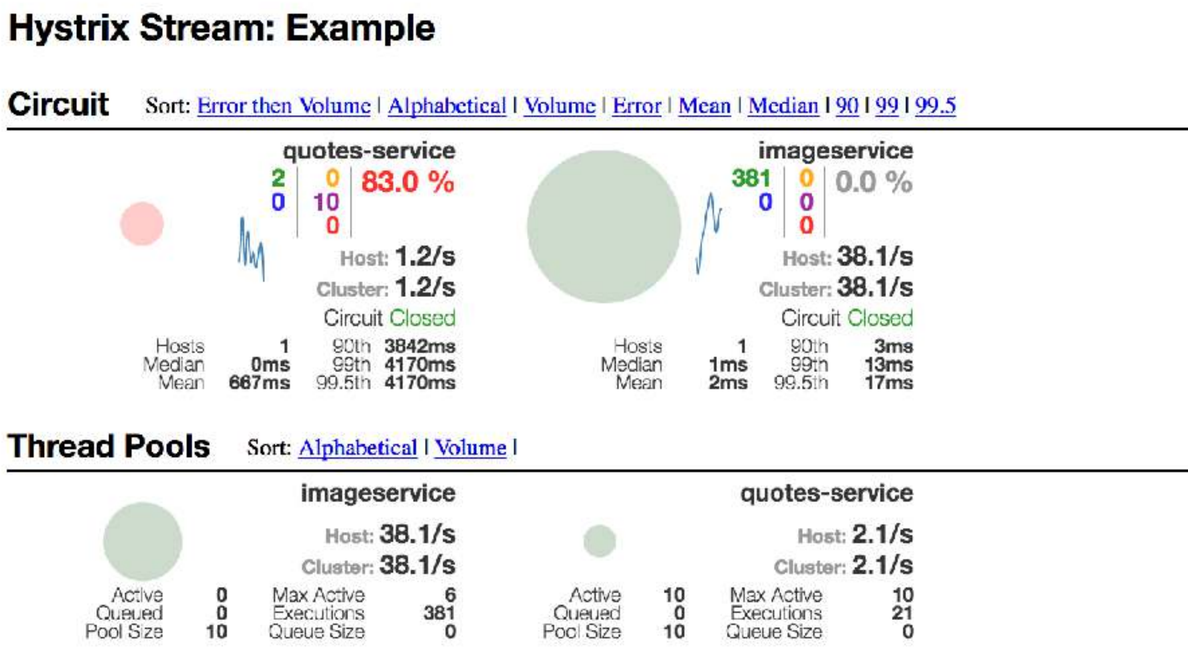


Cloud native stack

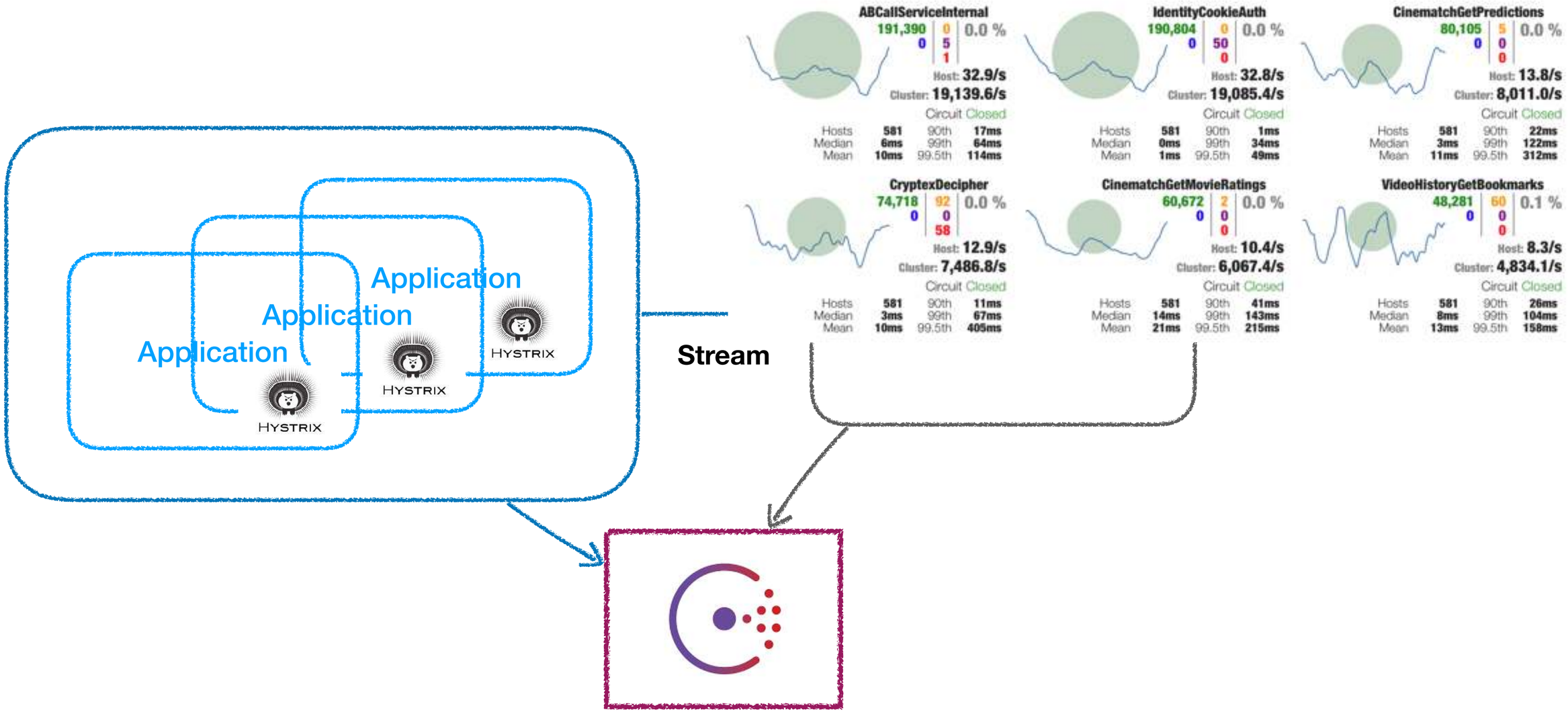


Hystrix

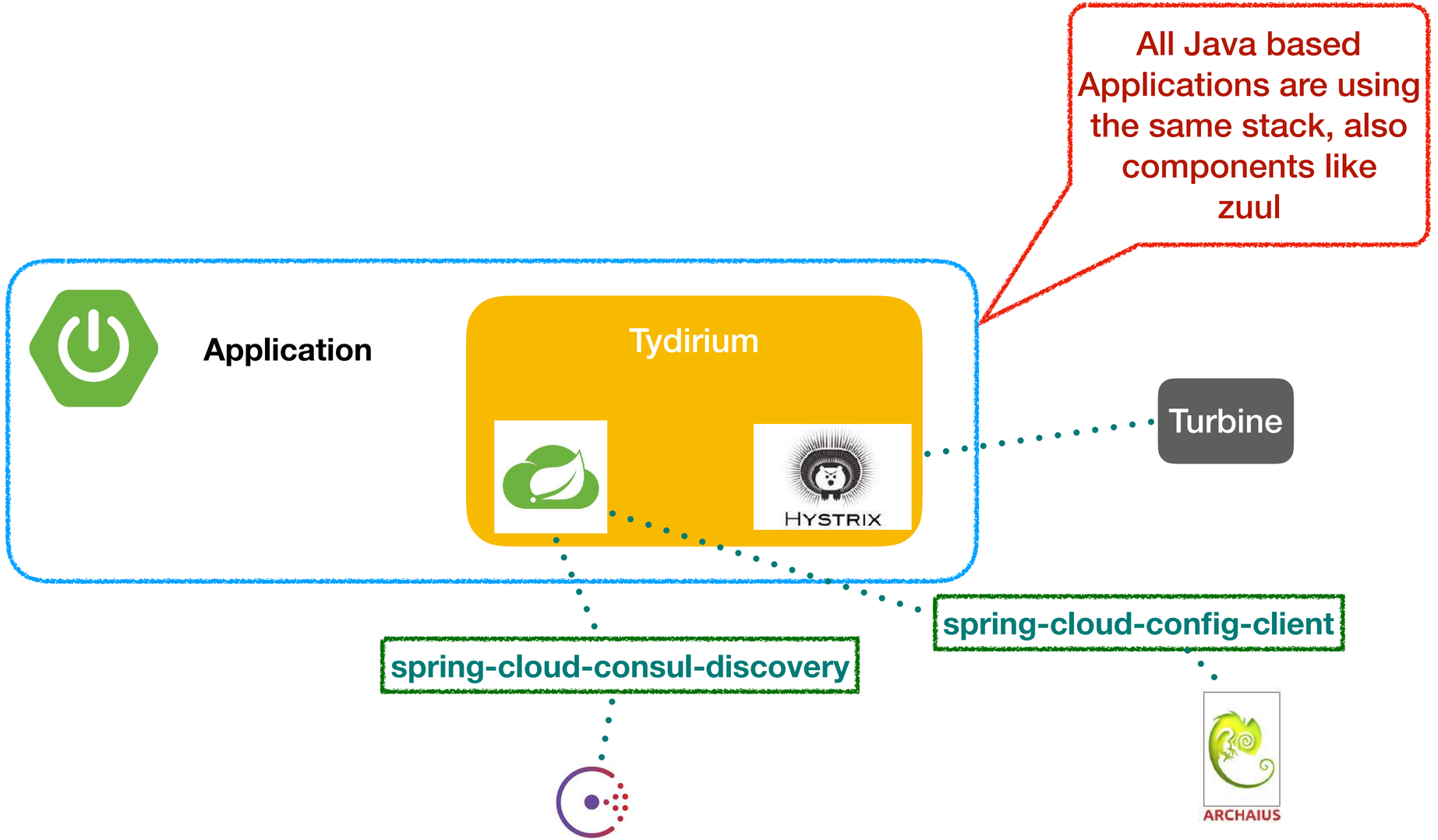

call / raise circuit open
success
fail [threshold reached]
Closed
Open
reset timeout
fail [under threshold]
fail
success
Half Open


Hystrix Stream: Example
Circuit
Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.5
quotes-service
2 0 83.0 %
0 10 0
Host: 1.2/s
Cluster: 1.2/s
Circuit Closed
Hosts 1
Median 0ms
Mean 667ms
90th 3842ms
99th 4170ms
99.5th 4170ms
imageservice
381 0 0.0 %
0 0 0
Host: 38.1/s
Cluster: 38.1/s
Circuit Closed
Hosts 1
Median 1ms
Mean 2ms
90th 3ms
99th 13ms
99.5th 17ms
Thread Pools
Sort: Alphabetical | Volume
imageservice
Host: 38.1/s
Cluster: 38.1/s
Active 0
Queued 0
Pool Size 10
Max Active 6
Executions 381
Queue Size 0
quotes-service
Host: 2.1/s
Cluster: 2.1/s
Active 10
Queued 0
Pool Size 10
Max Active 10
Executions 21
Queue Size 0

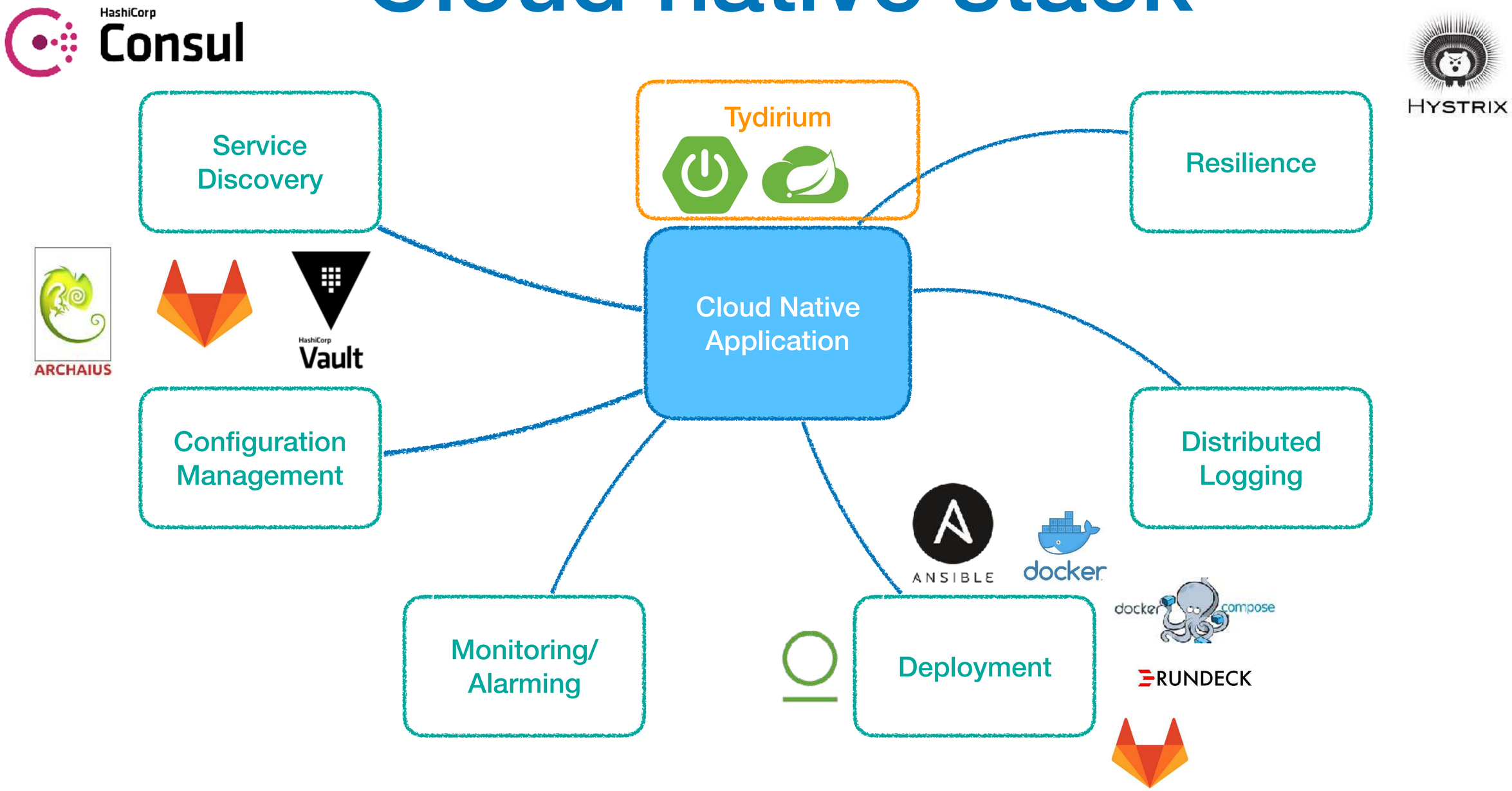
Resilience


ABCallServiceInternal
191,390 0 0.0 %
0 5
1
Host: 32.9/s
Cluster: 19,139.6/s
Circuit Closed
Hosts 581 90th 17ms
Median 6ms 99th 64ms
Mean 10ms 99.5th 114ms
IdentityCookieAuth
190,804 0 0.0 %
0 50
0
Host: 32.8/s
Cluster: 19,085.4/s
Circuit Closed
Hosts 581 90th 1ms
Median 0ms 99th 34ms
Mean 1ms 99.5th 49ms
CinematchGetPredictions
80,105 5 0.0 %
0 0
0
Host: 13.8/s
Cluster: 8,011.0/s
Circuit Closed
Hosts 581 90th 22ms
Median 3ms 99th 122ms
Mean 11ms 99.5th 312ms
CryptexDecipher
74,718 92 0.0 %
0 0
58
Host: 12.9/s
Cluster: 7,486.8/s
Circuit Closed
Hosts 581 90th 11ms
Median 3ms 99th 67ms
Mean 10ms 99.5th 405ms
CinematchGetMovieRatings
60,672 2 0.0 %
0 0
0
Host: 10.4/s
Cluster: 6,067.4/s
Circuit Closed
Hosts 581 90th 41ms
Median 14ms 99th 143ms
Mean 21ms 99.5th 215ms
VideoHistoryGetBookmarks
48,281 60 0.1 %
0 0
0
Host: 8.3/s
Cluster: 4,834.1/s
Circuit Closed
Hosts 581 90th 26ms
Median 8ms 99th 104ms
Mean 13ms 99.5th 158ms
Application
Application
Application
HYSTRIX
HYSTRIX
HYSTRIX
Stream

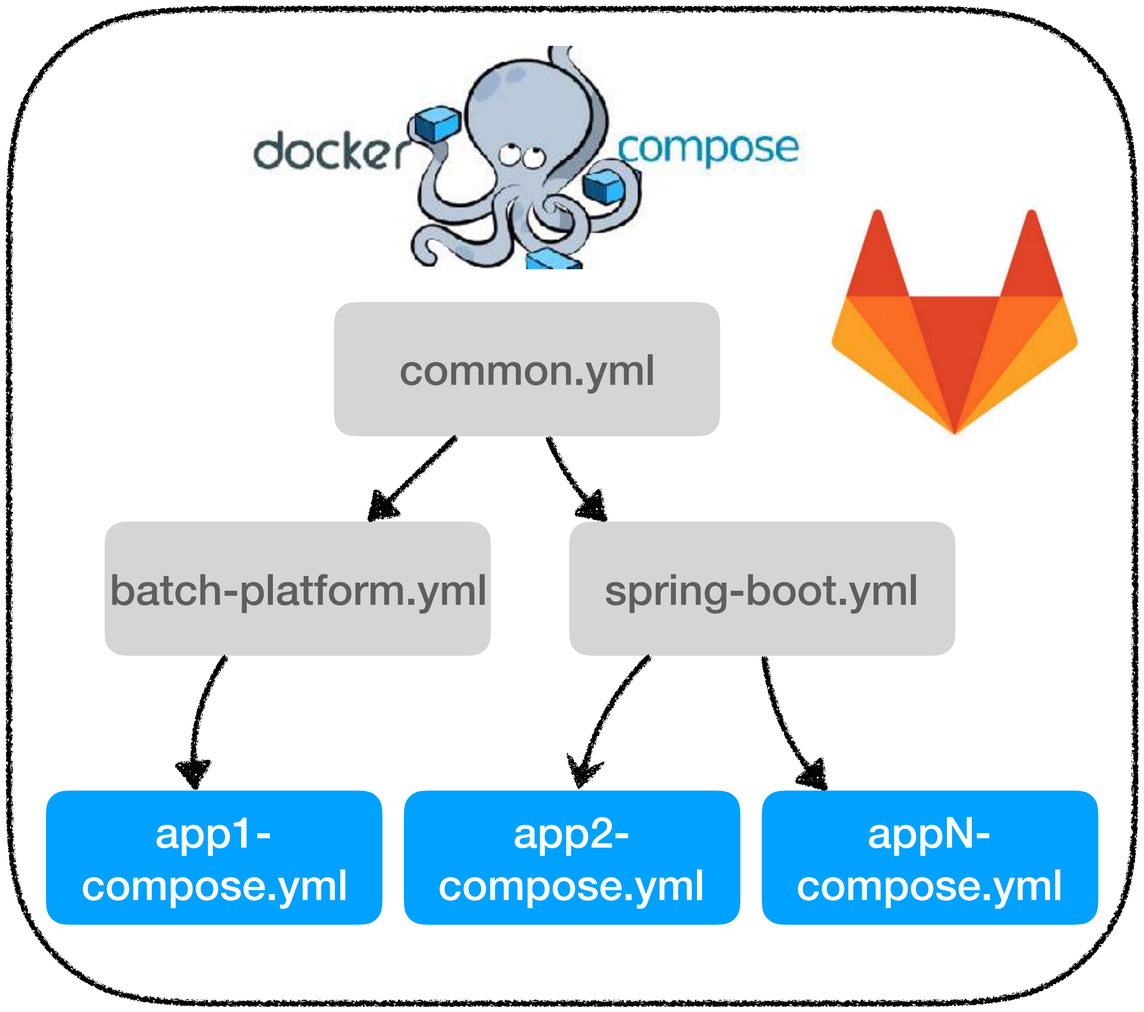
Resilience


All Java based Applications are using the same stack, also components like zuul
Application
Tydirium
HYSTRIX
Turbine
spring-cloud-consul-discovery
spring-cloud-config-client
ARCHAIUS

Cloud native stack



Docker compose

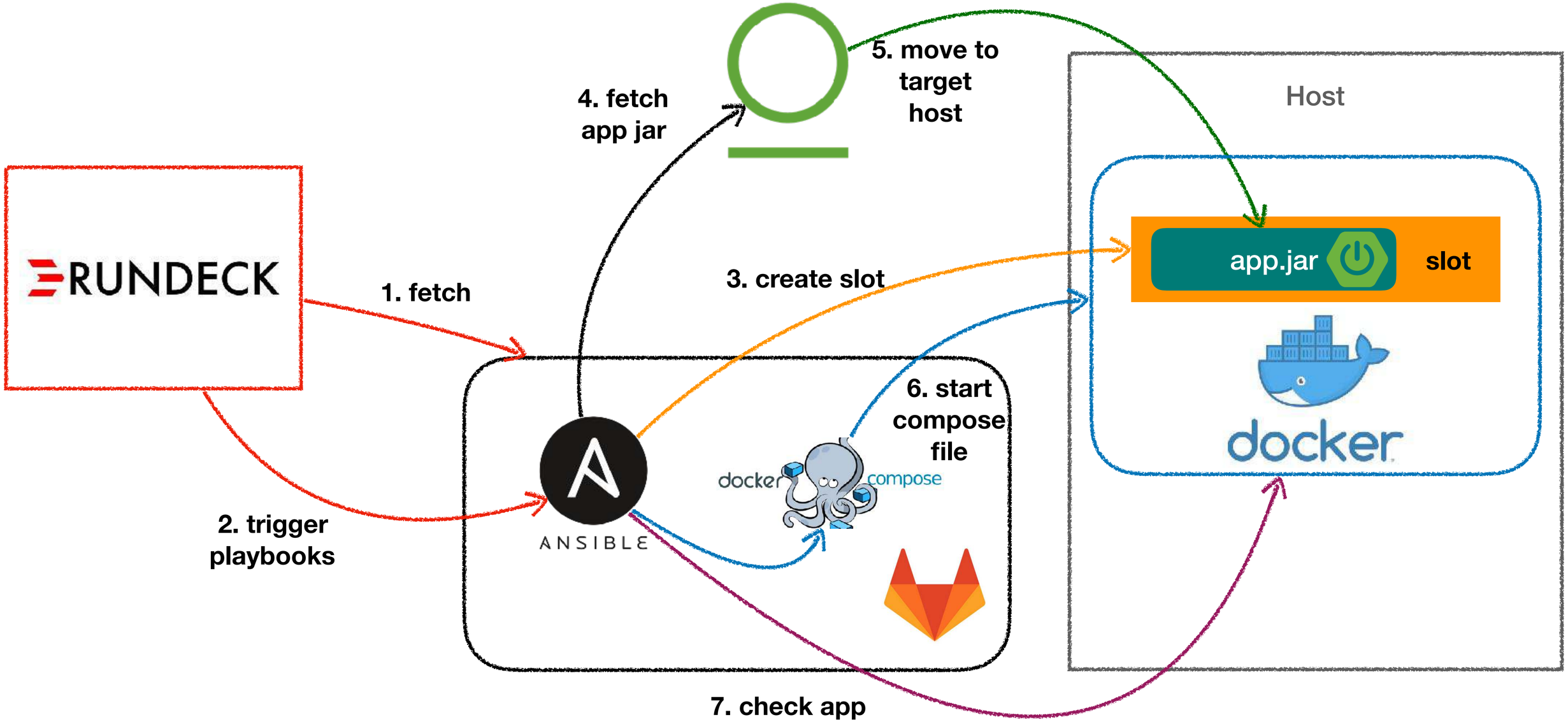



```
# The Consul connection will always go to the Consul (Agent/Server) running on the same host
SPRING_CLOUD_CONSUL_HOST: $SERVER_FQDN
SPRING_CLOUD_CONSUL_PORT: "8500"

# This enabled registering itself at Consul as a service
SPRING_CLOUD_CONSUL_DISCOVERY_HOSTNAME: $SERVER_FQDN
SPRING_CLOUD_CONSUL_DISCOVERY_SERVICE_NAME: "${spring.application.name}"
SPRING_CLOUD_CONSUL_DISCOVERY_INSTANCE_ID: "${spring.application.name}"

# This enables to load the config from the Spring Cloud Config Server
SPRING_CLOUD_CONFIG_DISCOVERY_ENABLED: "true"
SPRING_CLOUD_CONFIG_TOKEN: "${VAULT_TOKEN}"
SPRING_CLOUD_CONFIG_DISCOVERY_SERVICE_ID: tydirium-config-server
SPRING_CLOUD_CONFIG_FAIL_FAST: "true"

# This will propagte the version info over the /info endpoint to the Spring Boot Admin
INFO_version: $version
```

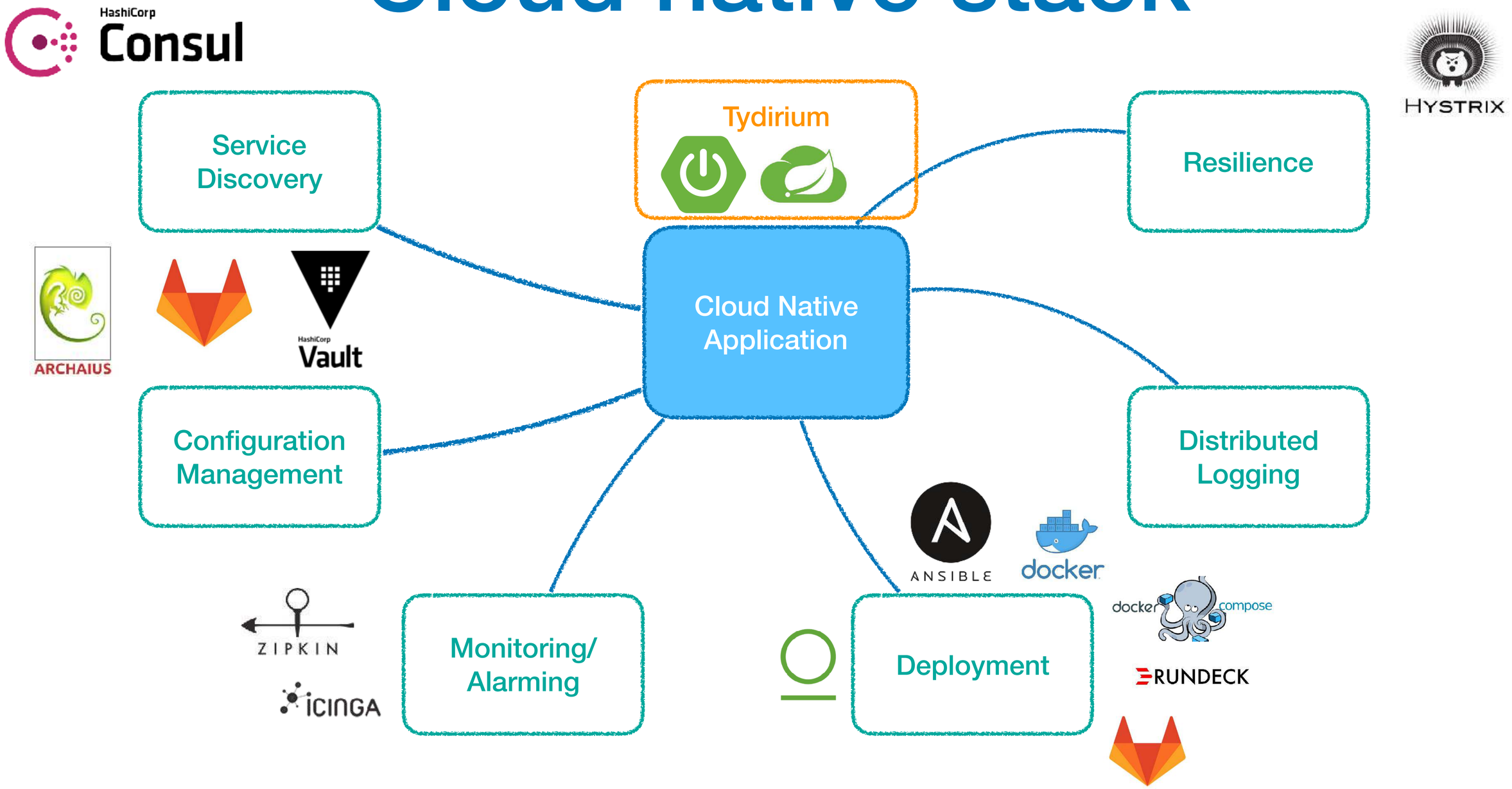
```
labels:
  ansible.healthcheck_port: "8080"
  ansible.healthcheck_path: "/health"
  ansible.healthcheck_json_property: "status"
  ansible.healthcheck_json_value: "UP"
  icinga.extended_healthcheck_path: "/extended-health"
  ansible.target_filename: app
  ansible.artifact_extension: jar
```

```
version: '2.4'
services:

  hope-facade:
    extends:
      file: templates/docker-compose.springboot.yml
      service: springboot
    container_name: hope-facade
    volumes:
      - /home/$SERVER_USER/server/hope-facade:/usr/local/springboot
    environment:
      SPRING_APPLICATION_NAME: hope-facade
      SPRING_CLOUD_CONSUL_DISCOVERY_PORT: "9080"
    labels:
      ansible.group_id: com.unitedinternet.buizsol
      ansible.artifact_id: hope-facade
    ports:
      - "9080:8080"
      - "8010:8000"
      - "20010:20000"
```

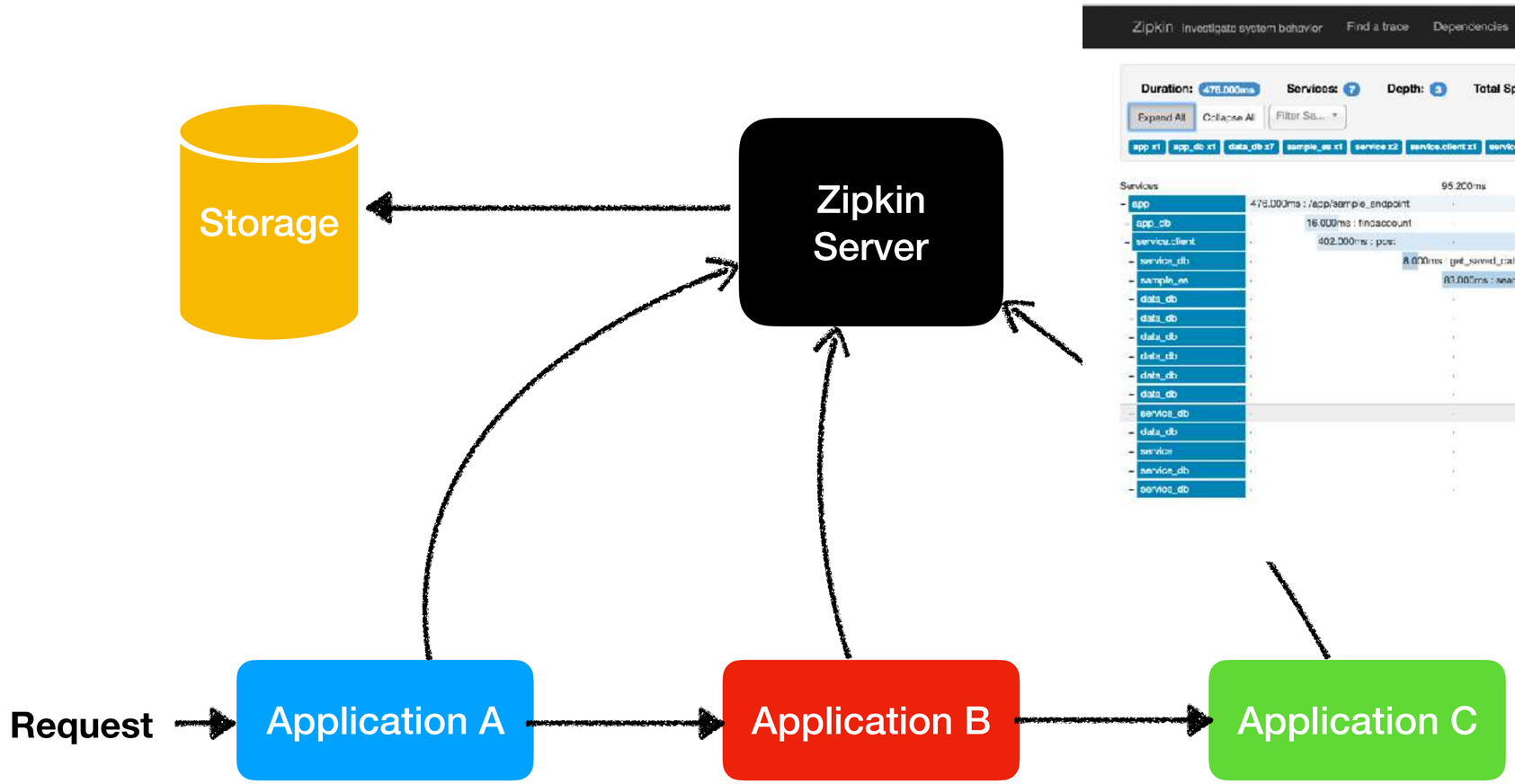
Deployment

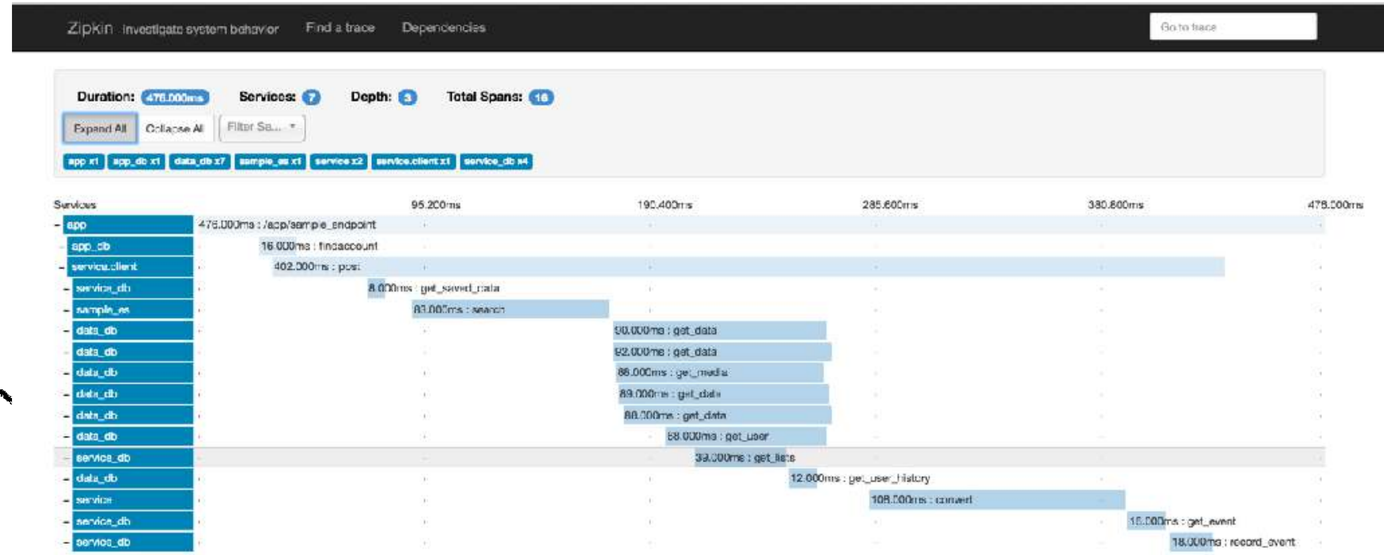

RUNDECK
1. fetch
2. trigger playbooks
ANSIBLE
3. create slot
4. fetch app jar
5. move to target host
docker compose
6. start compose file
7. check app
Host
app.jar
slot
docker

Cloud native stack



Distributed tracing

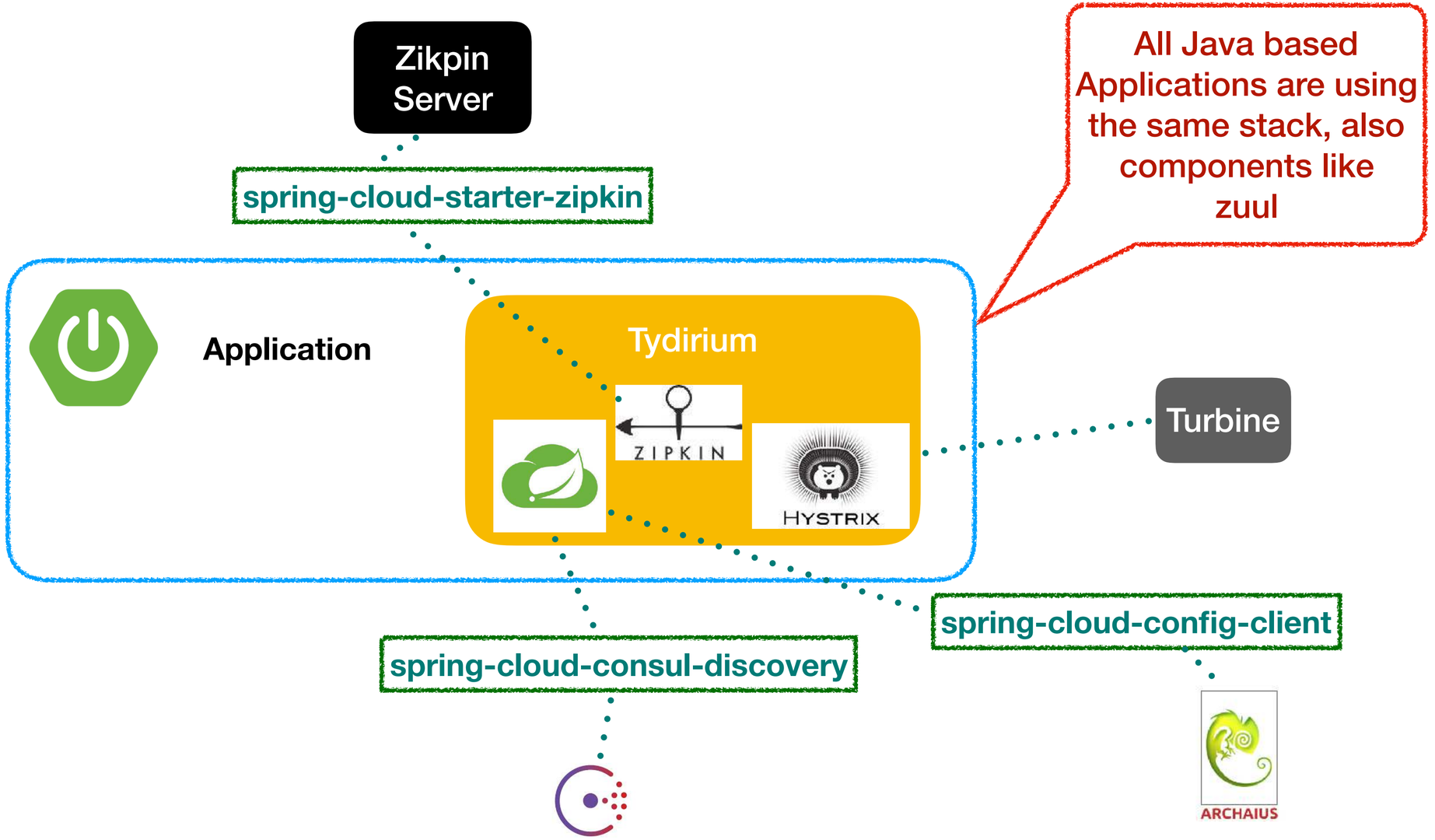




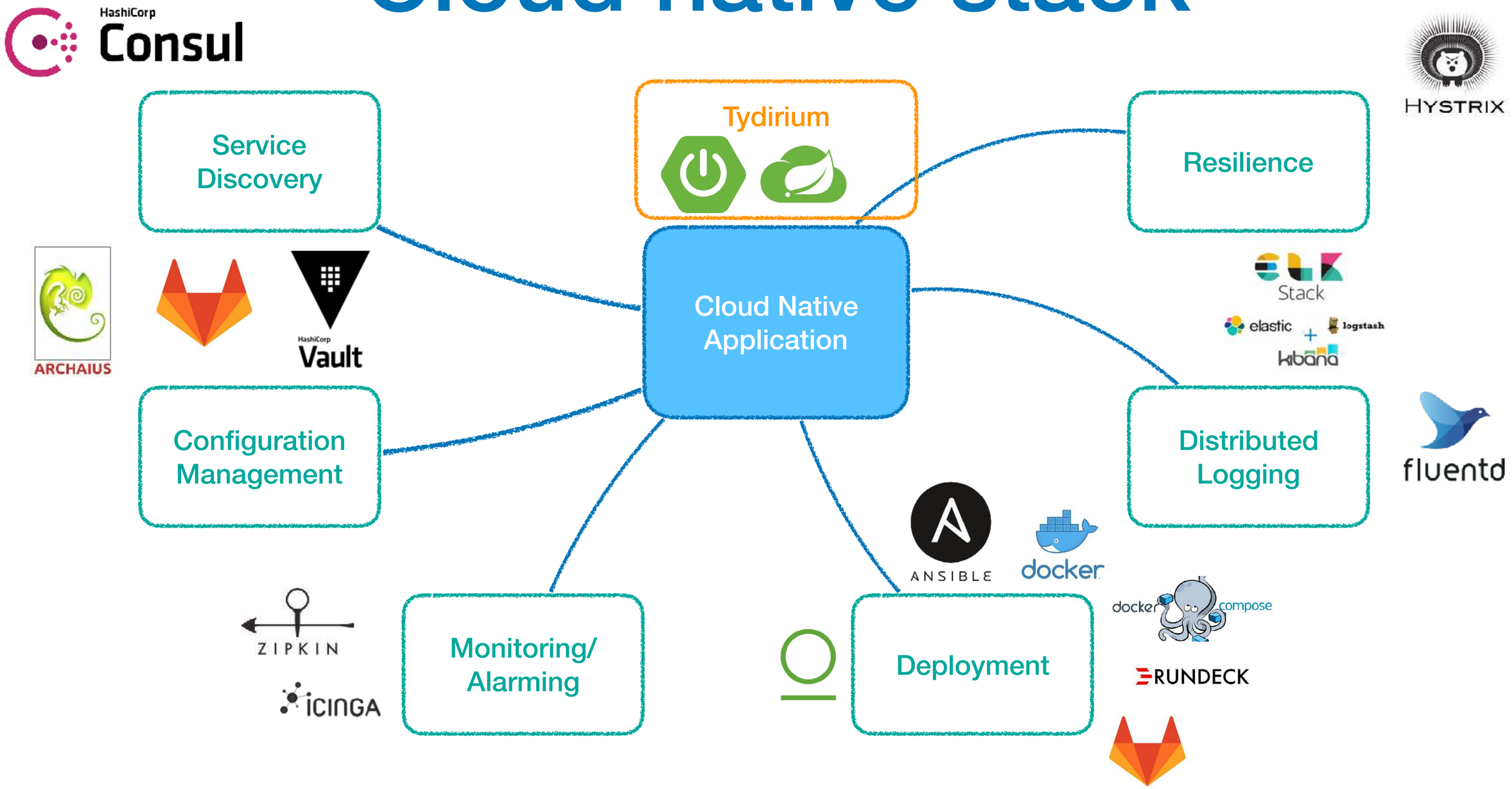
Monitoring & alarming

- Inciga2 configurations automatically created on Deployment
- Other monitoring tools:
  - Spring Boot Admin
  - Microservice Dashboard

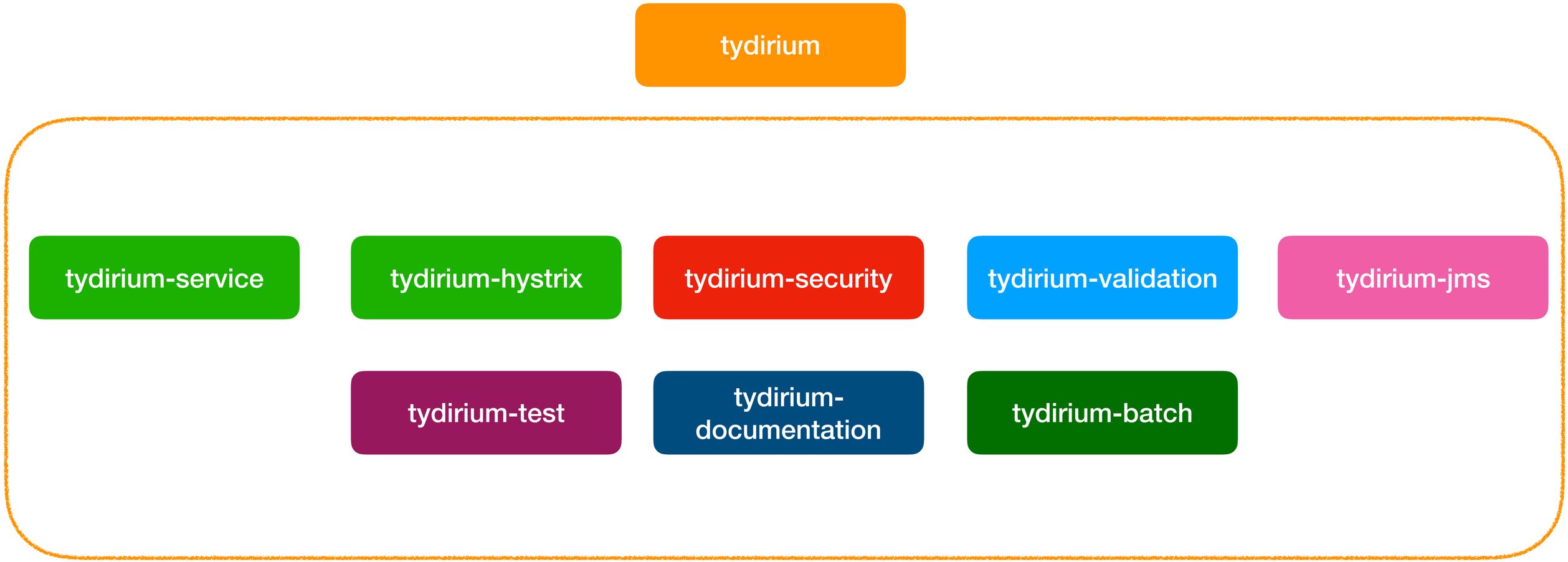
Distributed tracing



Cloud native stack



Tydirium


tydirium
tydirium-service
tydirium-hystrix
tydirium-security
tydirium-validation
tydirium-jms
tydirium-test
tydirium-documentation
tydirium-batch

What else?

- Cloud Native Batch Platform
  - Spring Batch, Spring Batch Lightmin, Tydirium
- Cloud Native Process Platform
  - Camunda BPMN & Tydirium

Where we want to go

Next steps planned

- Fully-automated HW provisioning
- QA automation
- Distributed session management
- Container platform (e.g. K8s, Nomad)
- Deploy pipelines (e.g. GoCD)


Q
&
A