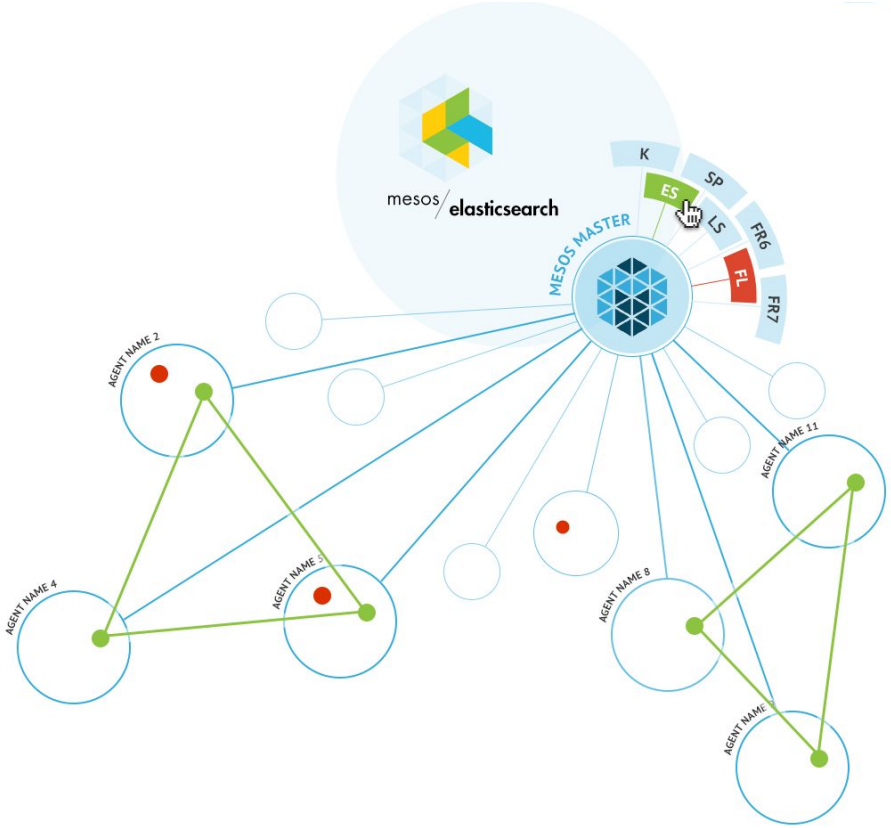
ContainerSolutions

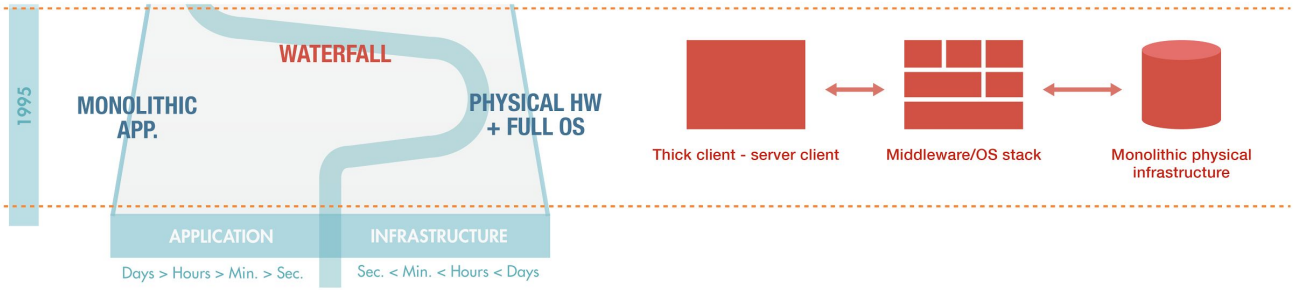
Testing Cloud Native Applications

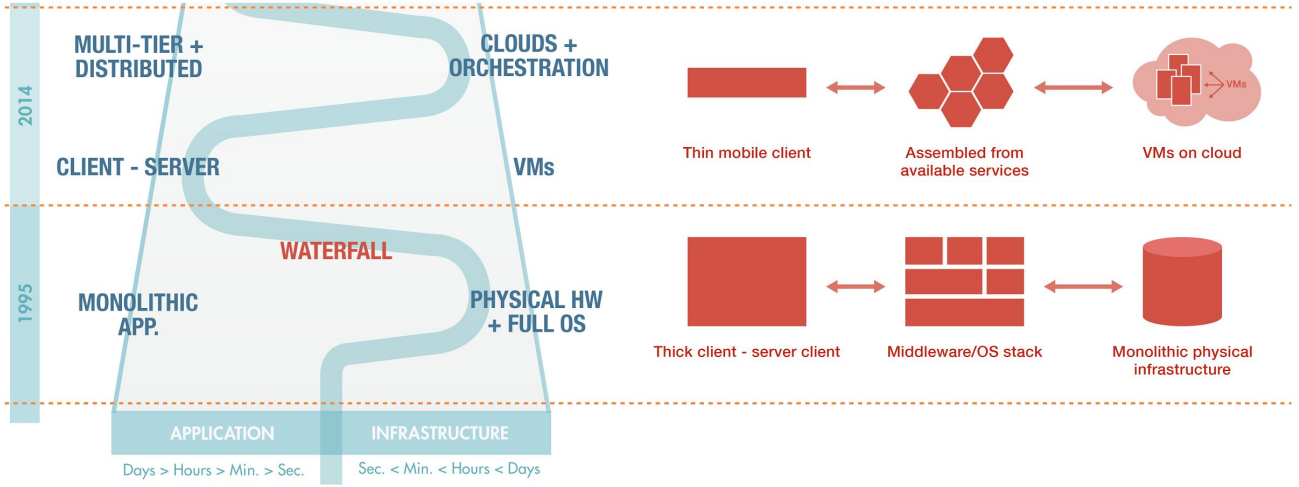
Pini Reznik, **@pini42**

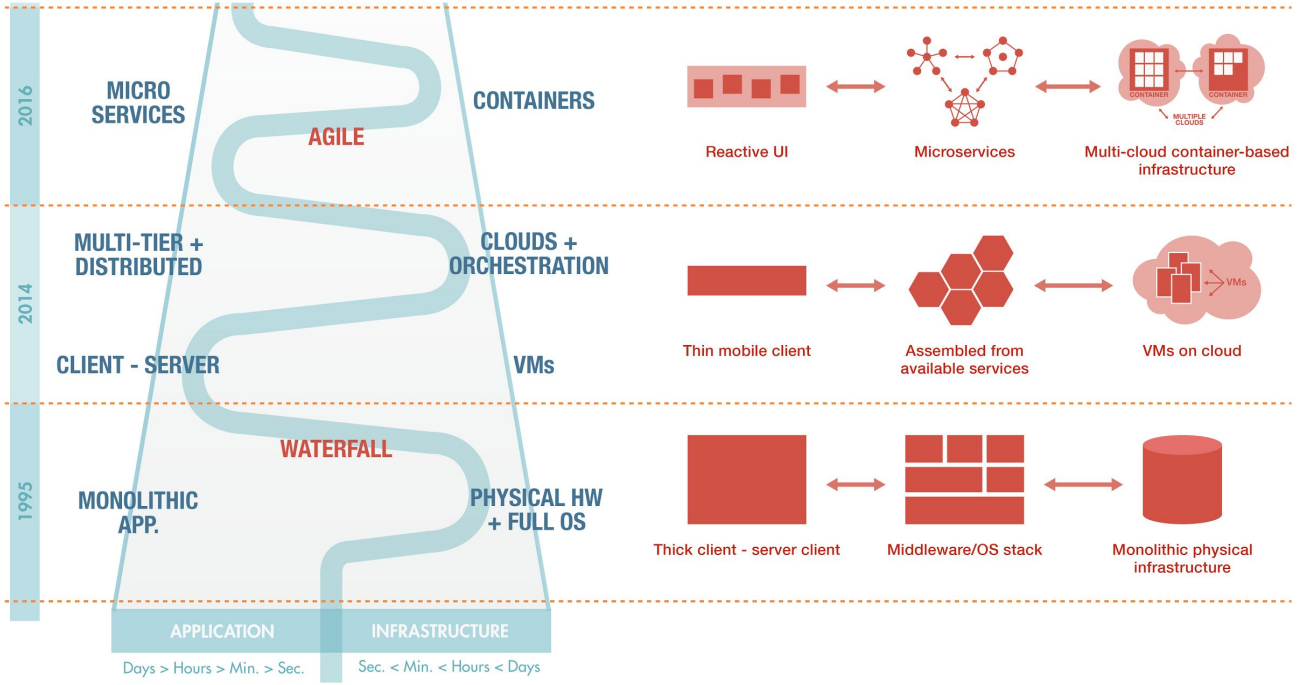
October 2016

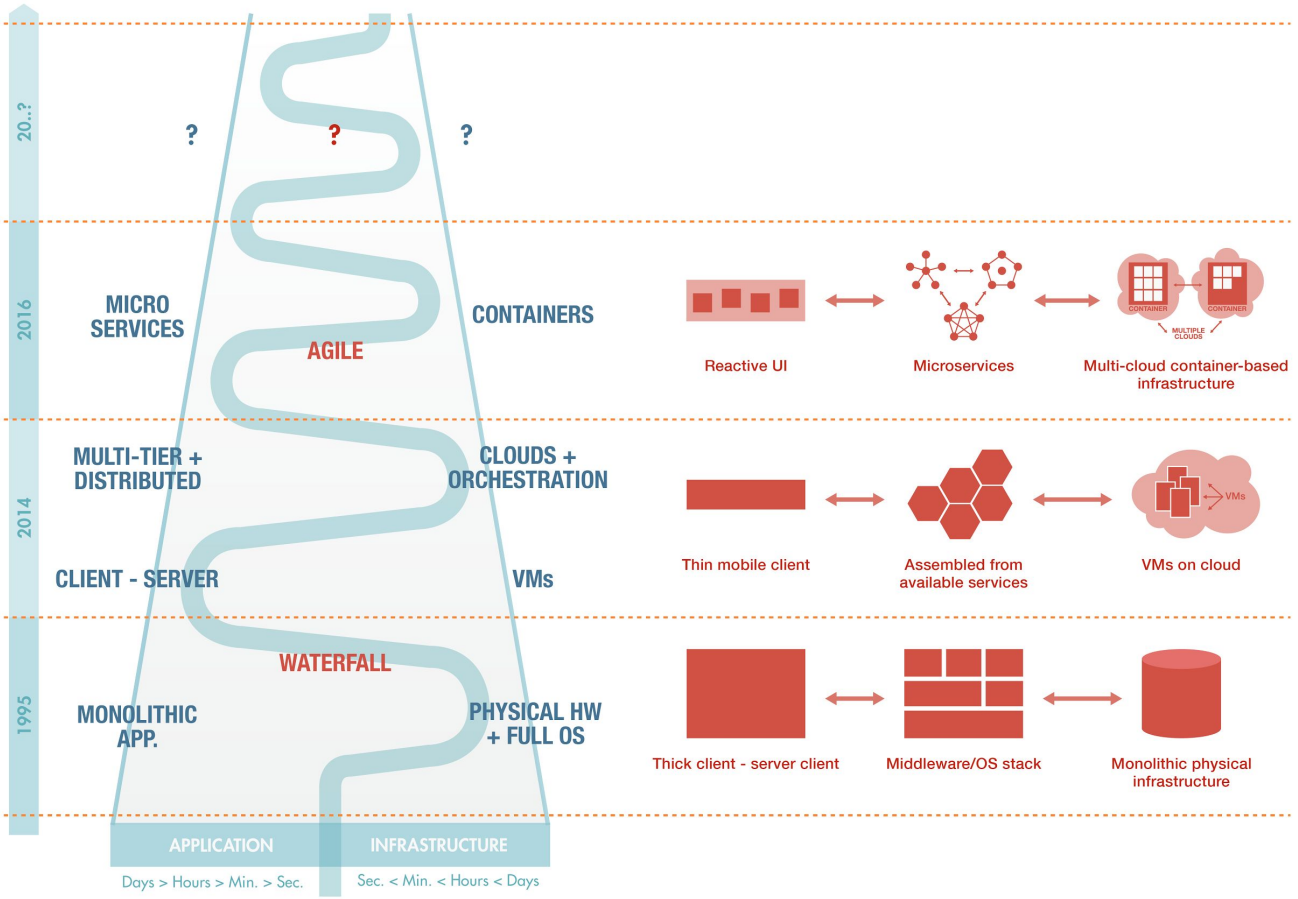


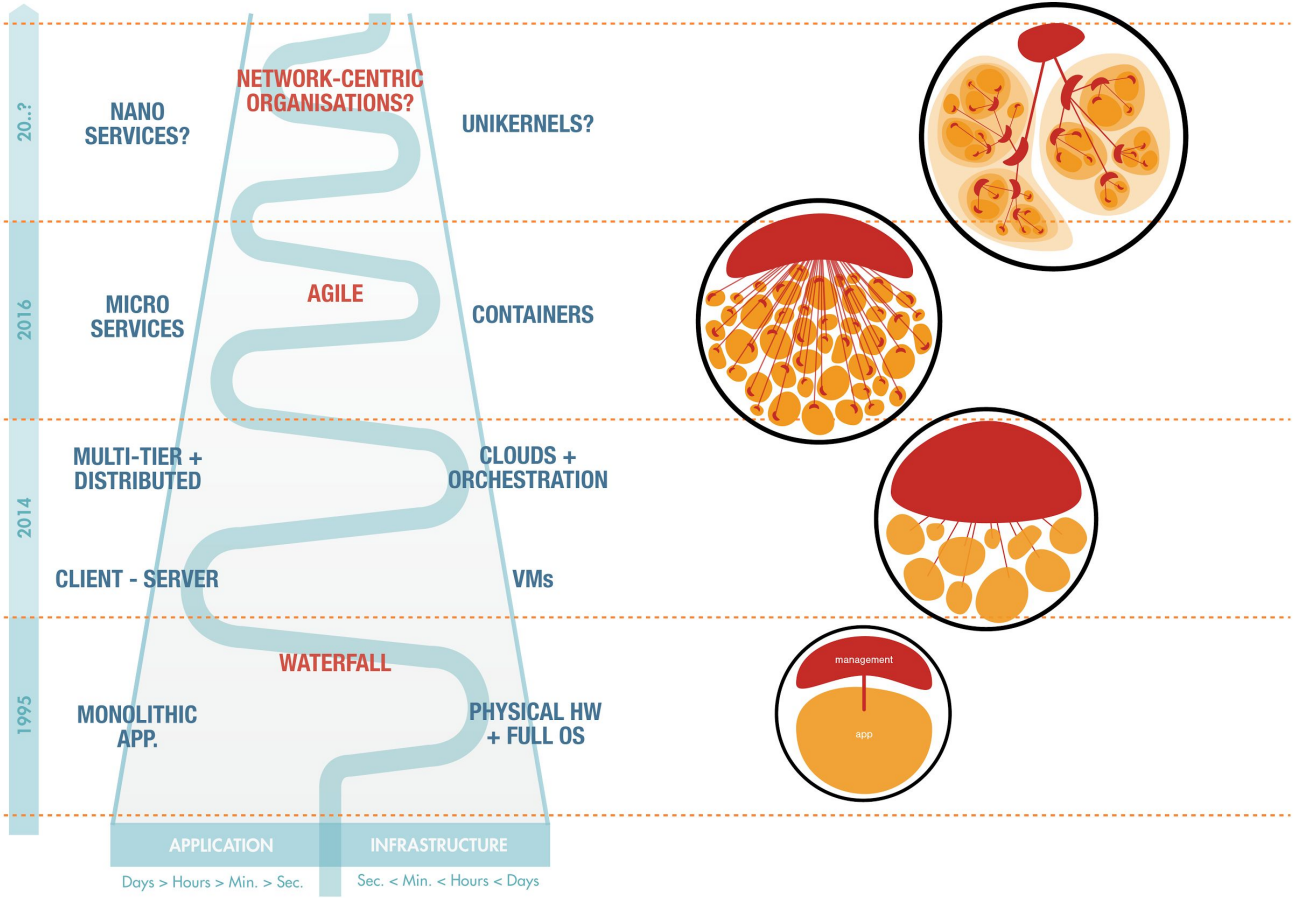
Some background...


1995
WATERFALL
MONOLITHIC APP.
PHYSICAL HW + FULL OS
APPLICATION
INFRASTRUCTURE
Days > Hours > Min. > Sec.
Sec. < Min. < Hours < Days
Thick client - server client
Middleware/OS stack
Monolithic physical infrastructure

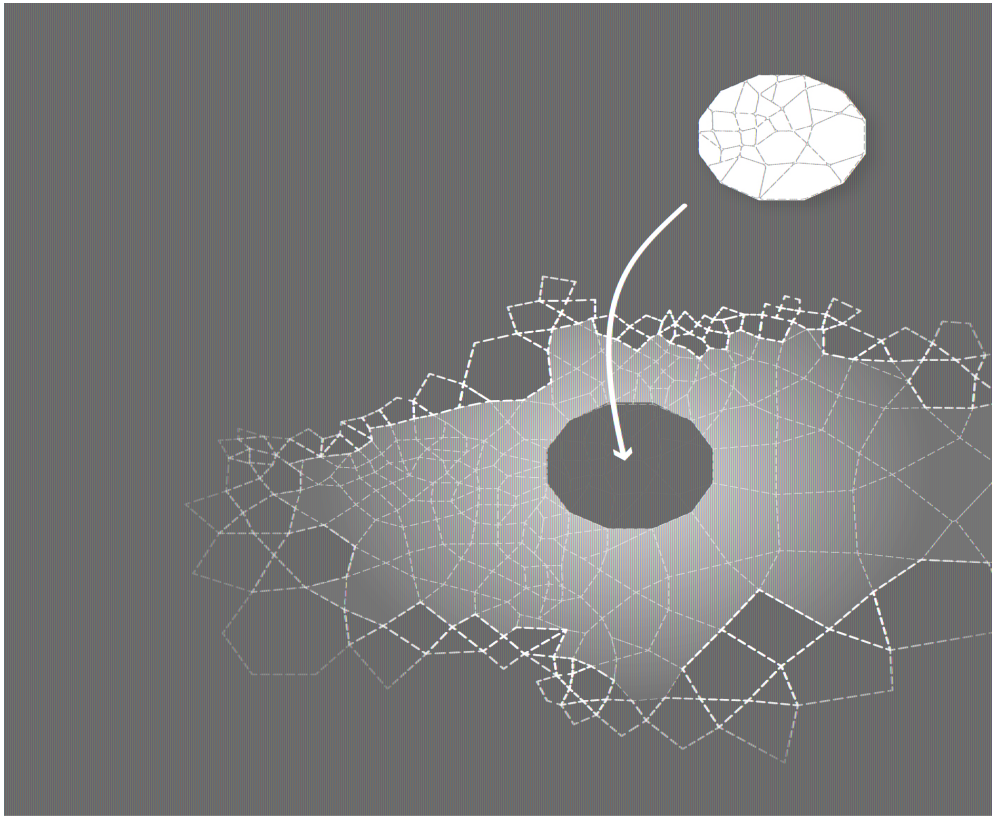

2014
1995
MULTI-TIER + DISTRIBUTED
CLOUDS + ORCHESTRATION
CLIENT - SERVER
VMs
WATERFALL
MONOLITHIC APP.
PHYSICAL HW + FULL OS
APPLICATION
INFRASTRUCTURE
Days > Hours > Min. > Sec.
Sec. < Min. < Hours < Days
Thin mobile client
Assembled from available services
VMs on cloud
Thick client - server client
Middleware/OS stack
Monolithic physical infrastructure

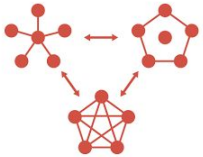

2016
2014
1995
MICRO SERVICES
AGILE
CONTAINERS
MULTI-TIER + DISTRIBUTED
CLOUDS + ORCHESTRATION
CLIENT - SERVER
VMs
WATERFALL
MONOLITHIC APP.
PHYSICAL HW + FULL OS
APPLICATION
INFRASTRUCTURE
Days > Hours > Min. > Sec.
Sec. < Min. < Hours < Days
Reactive UI
Microservices
CONTAINER
CONTAINER
MULTIPLE CLOUDS
Multi-cloud container-based infrastructure
Thin mobile client
Assembled from available services
VMs
VMs on cloud
Thick client - server client
Middleware/OS stack
Monolithic physical infrastructure


20..?
2016
2014
1995
?
?
?
MICRO SERVICES
AGILE
CONTAINERS
MULTI-TIER + DISTRIBUTED
CLOUDS + ORCHESTRATION
CLIENT - SERVER
VMs
WATERFALL
MONOLITHIC APP.
PHYSICAL HW + FULL OS
APPLICATION
INFRASTRUCTURE
Days > Hours > Min. > Sec.
Sec. < Min. < Hours < Days
Reactive UI
Microservices
CONTAINER
CONTAINER
MULTIPLE CLOUDS
Multi-cloud container-based infrastructure
Thin mobile client
Assembled from available services
VMs
VMs on cloud
Thick client - server client
Middleware/OS stack
Monolithic physical infrastructure


20..?
2016
2014
1995
NANO SERVICES?
NETWORK-CENTRIC ORGANISATIONS?
UNIKERNELS?
MICRO SERVICES
AGILE
CONTAINERS
MULTI-TIER + DISTRIBUTED
CLOUDS + ORCHESTRATION
CLIENT - SERVER
VMs
WATERFALL
MONOLITHIC APP.
PHYSICAL HW + FULL OS
management
app
APPLICATION
INFRASTRUCTURE
Days > Hours > Min. > Sec.
Sec. < Min. < Hours < Days

What is Cloud Native then?





Microservices

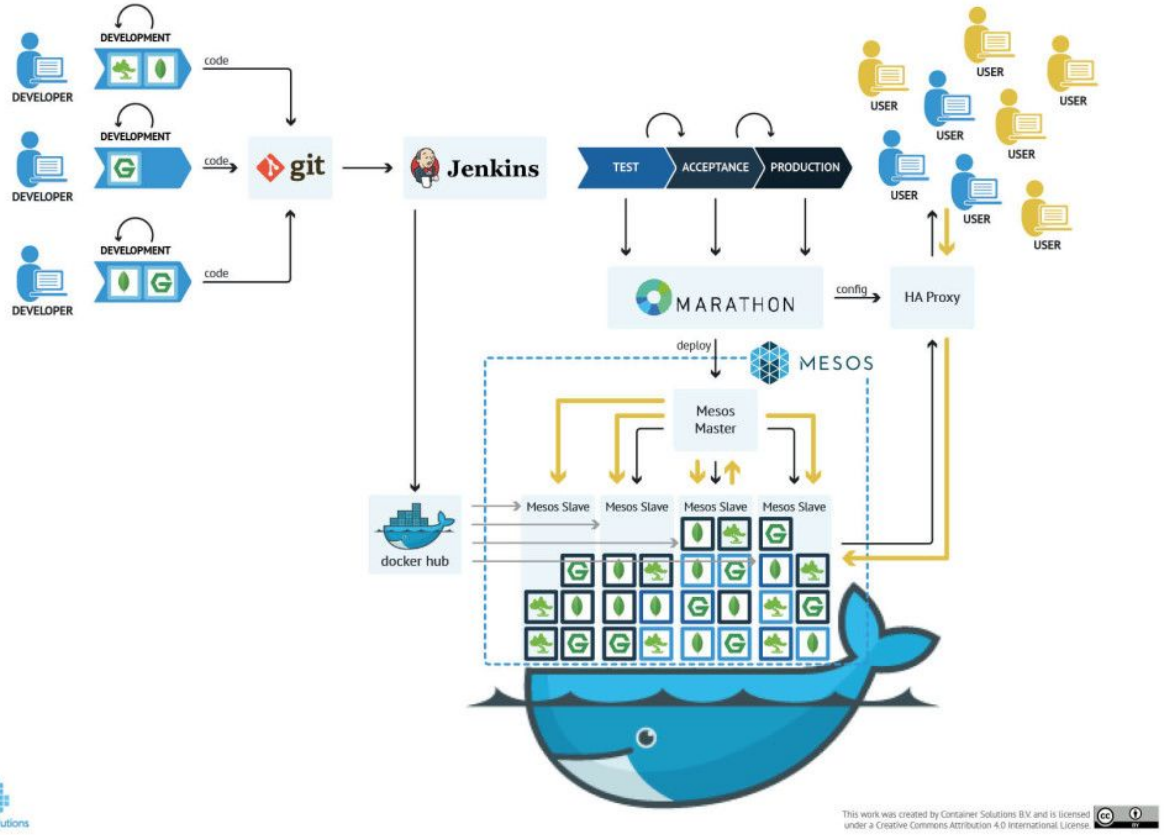
THE TWELVE-FACTOR APP

How CNCF defines its role?

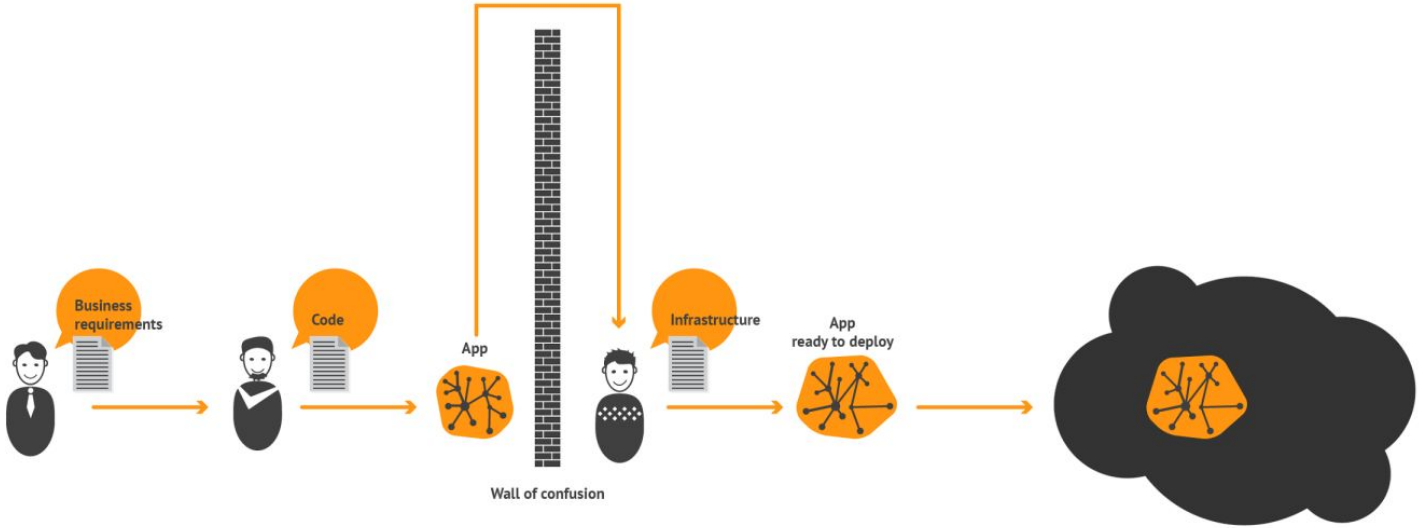
The Cloud Native Computing Foundation (CNCF) will harmonize emerging technologies and **foster innovation in container packaged, dynamically scheduled, and microservices based application development and operations.**

The mission of the CNCF is to create and drive the adoption of a new computing paradigm that is optimized for modern distributed systems environments capable of **scaling to tens of thousands of self healing multi-tenant nodes**.

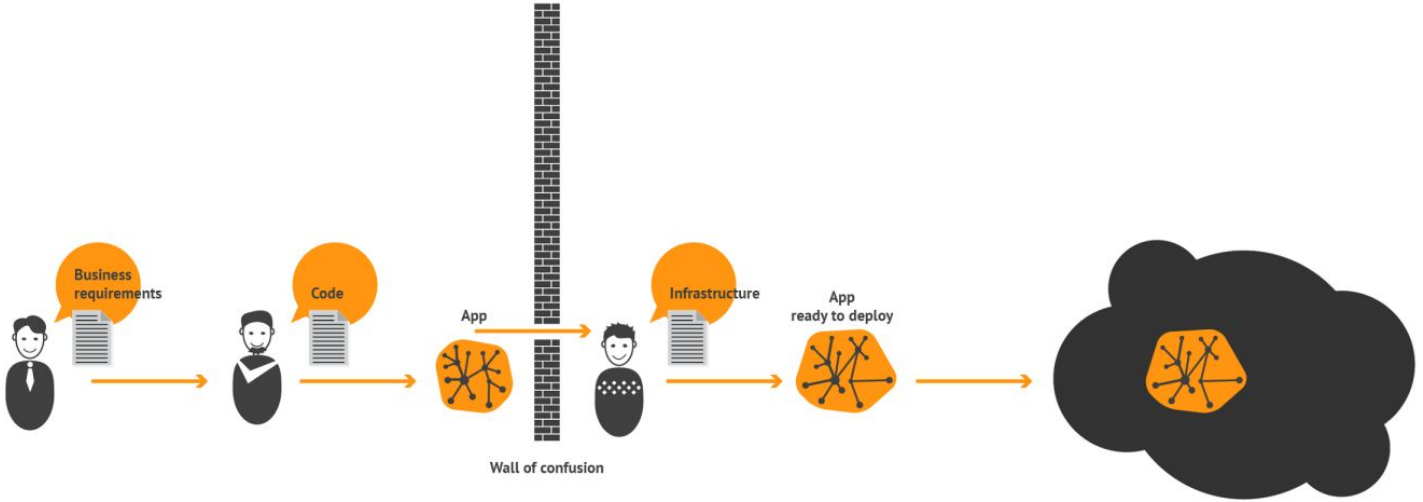
Continuous Delivery pipelines



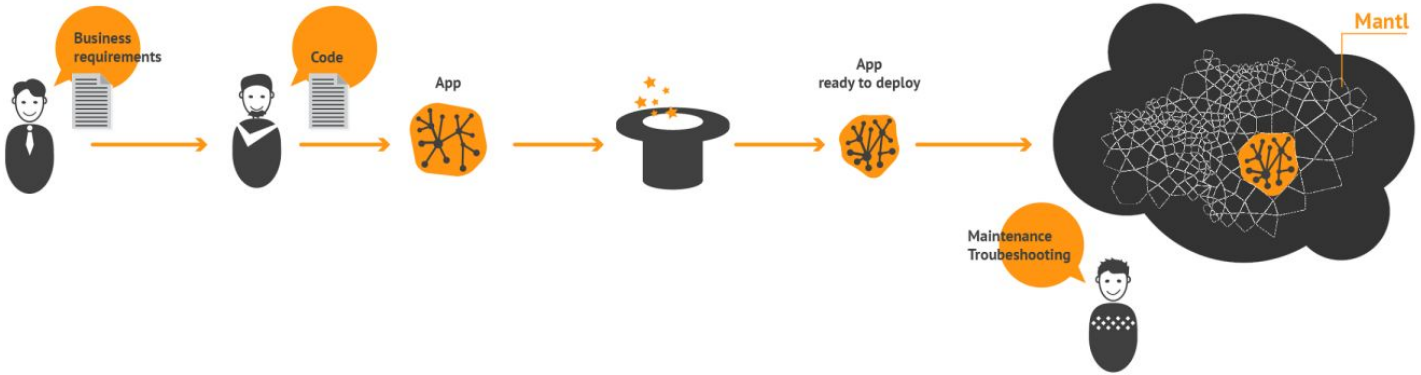
Pre-DevOps


Business requirements
Code
App
Wall of confusion
Infrastructure
App ready to deploy

DevOps


Business requirements
Code
App
Infrastructure
App ready to deploy
Wall of confusion

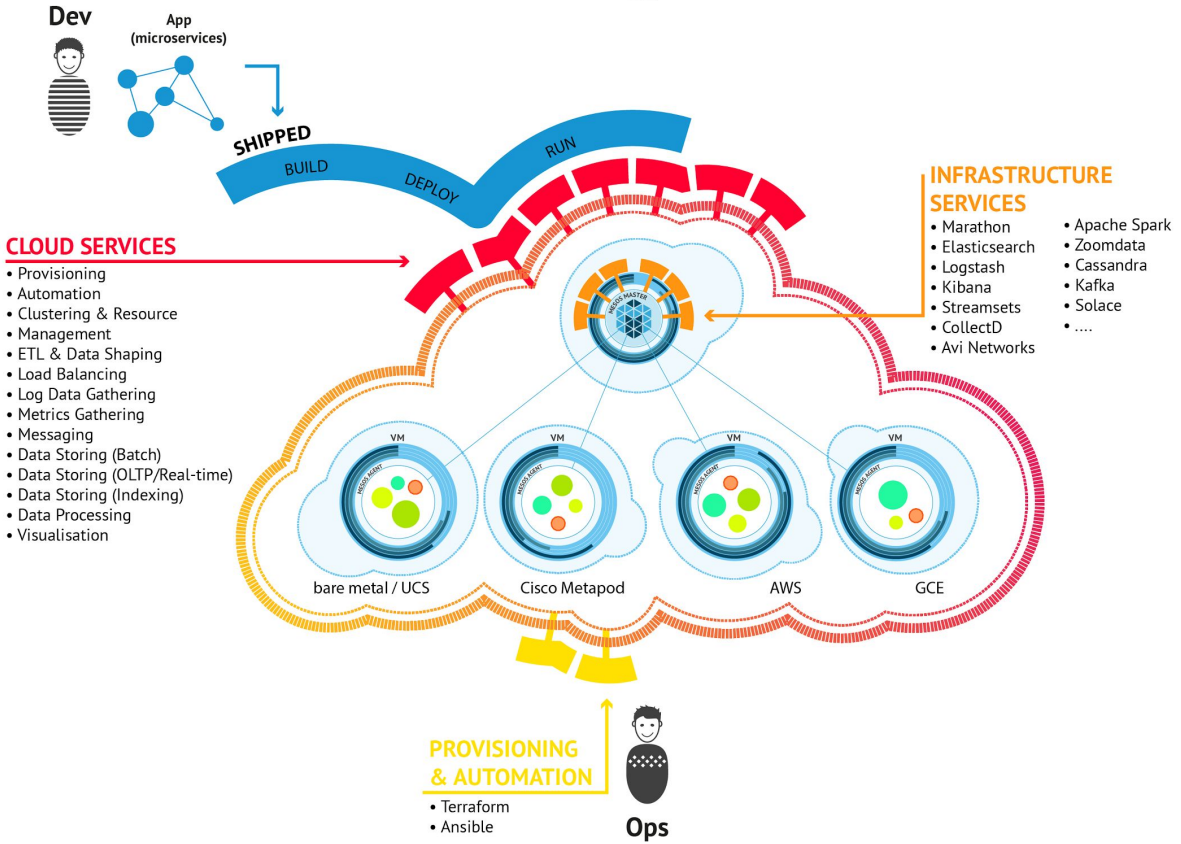
Post-DevOps


Business requirements
Code
App
App ready to deploy
Mantl
Maintenance Troubeshooting

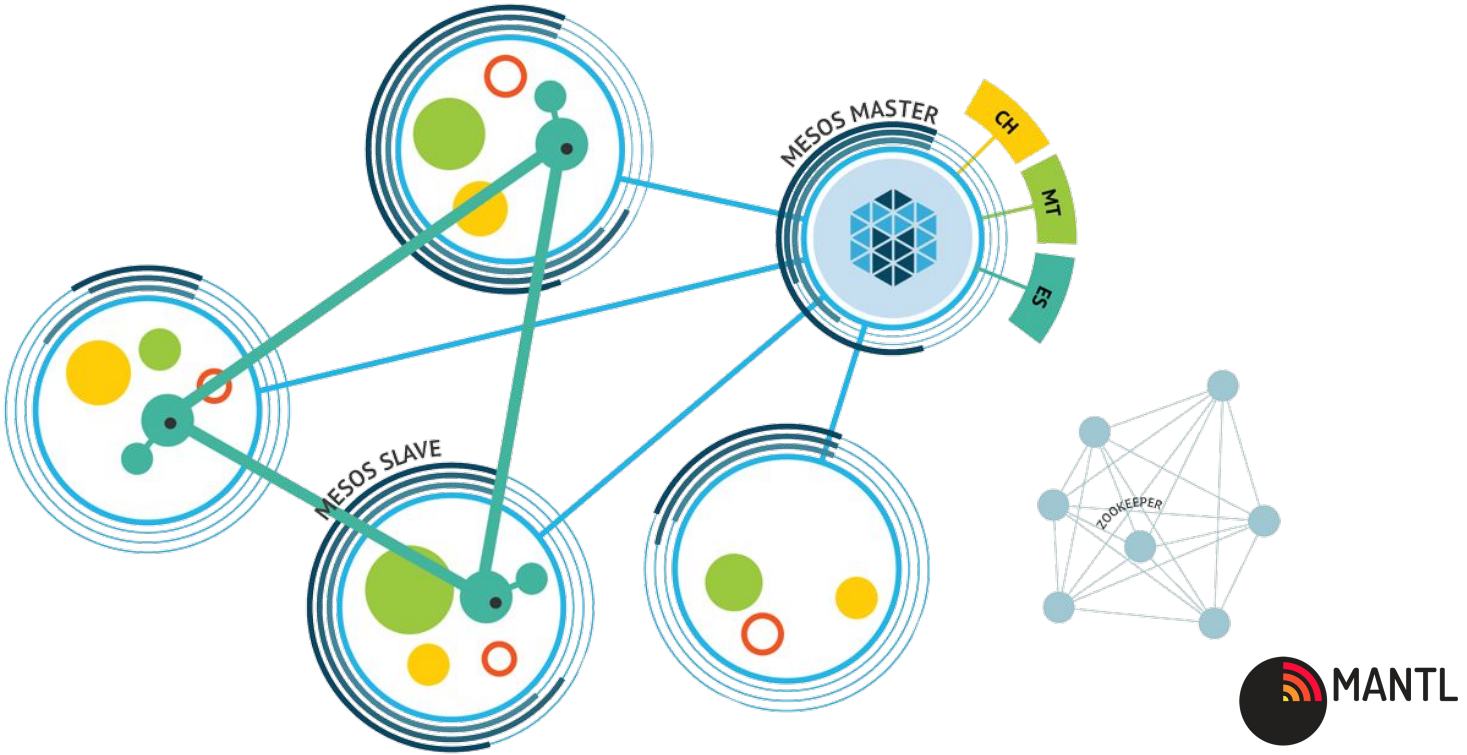
No manual testing as a principle!

Now *our* journey begins...

SHIPPED + MANTL



Building Elasticsearch framework for Mesos



Conclusion ...

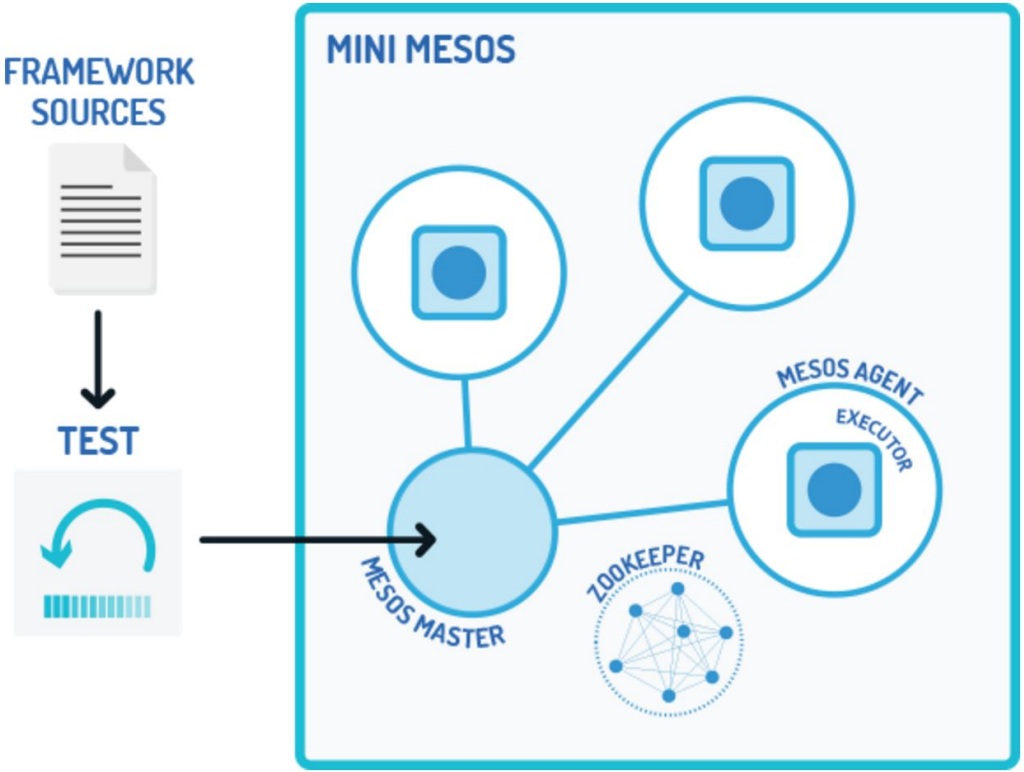
Mesos Frameworks Development is difficult!

We need a SDK

Possible Mesos Frameworks SDK

- HTTP APIs
- Framework template
- Rapid development environment
- Libraries with standard functionalities
- Testing tools
- Resilience/stress testing
- Distributed debugging
- Dependencies between frameworks (like Maven)
- Built-in infra services (networking, storage, security, load balancing, etc)
- Unified UI, including frameworks
- Documentation
- Workshops and online tutorials

Minimesos - First Steps



minimesos

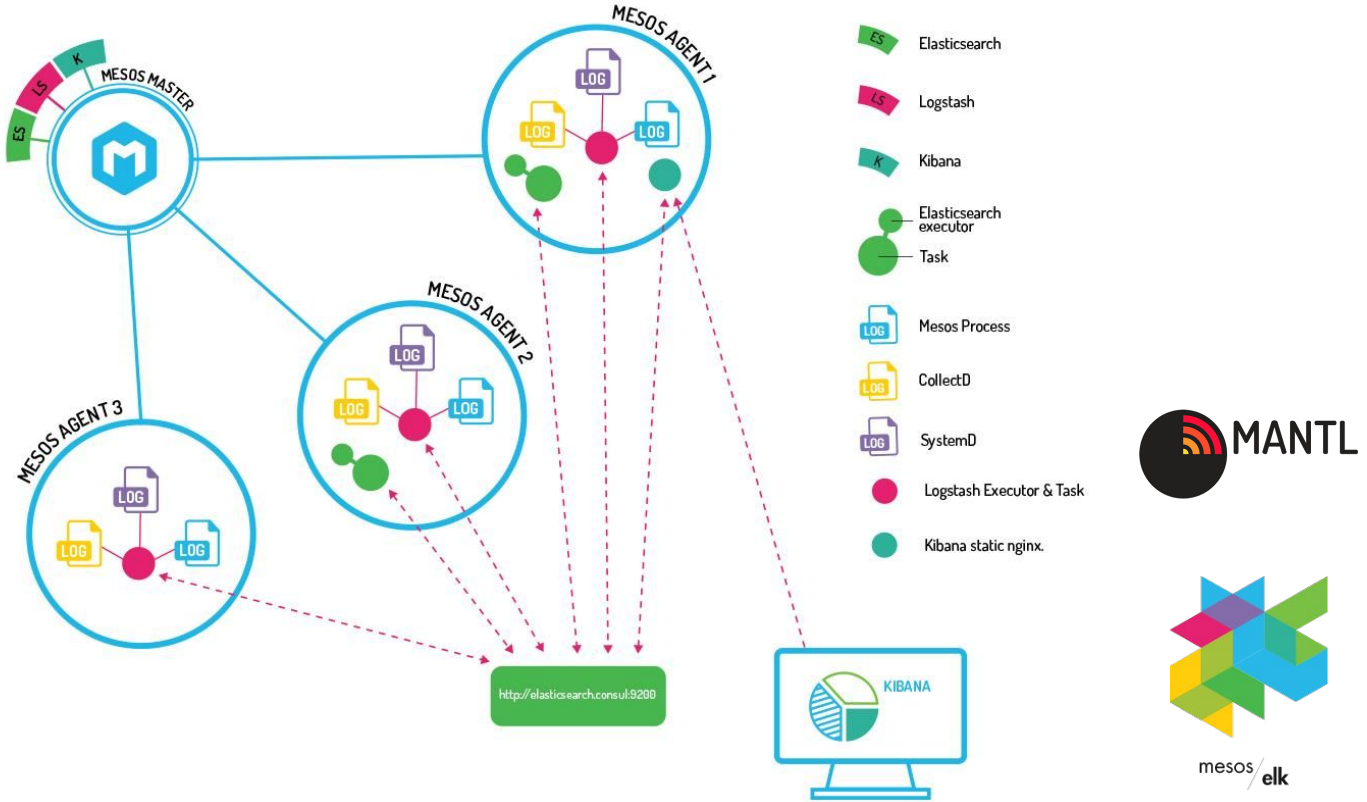
Minimesos - Java API

```
public class MesosClusterTest {

    @ClassRule
    public static MesosClusterTestRule testRule =

MesosClusterTestRule.fromFile("src/test/resources/configFiles/testMinimesosFile");

    public static MesosCluster cluster = testRule.getMesosCluster();

    @Test
    public void mesosClusterCanBeStarted() throws Exception {
        JSONObject stateInfo = cluster.getStateInfoJSON();
        Assert.assertEquals(3, stateInfo.getInt("activated_slaves"));
        Assert.assertTrue(cluster.getMesosMasterURL().contains(":5050"));
    }
}
```

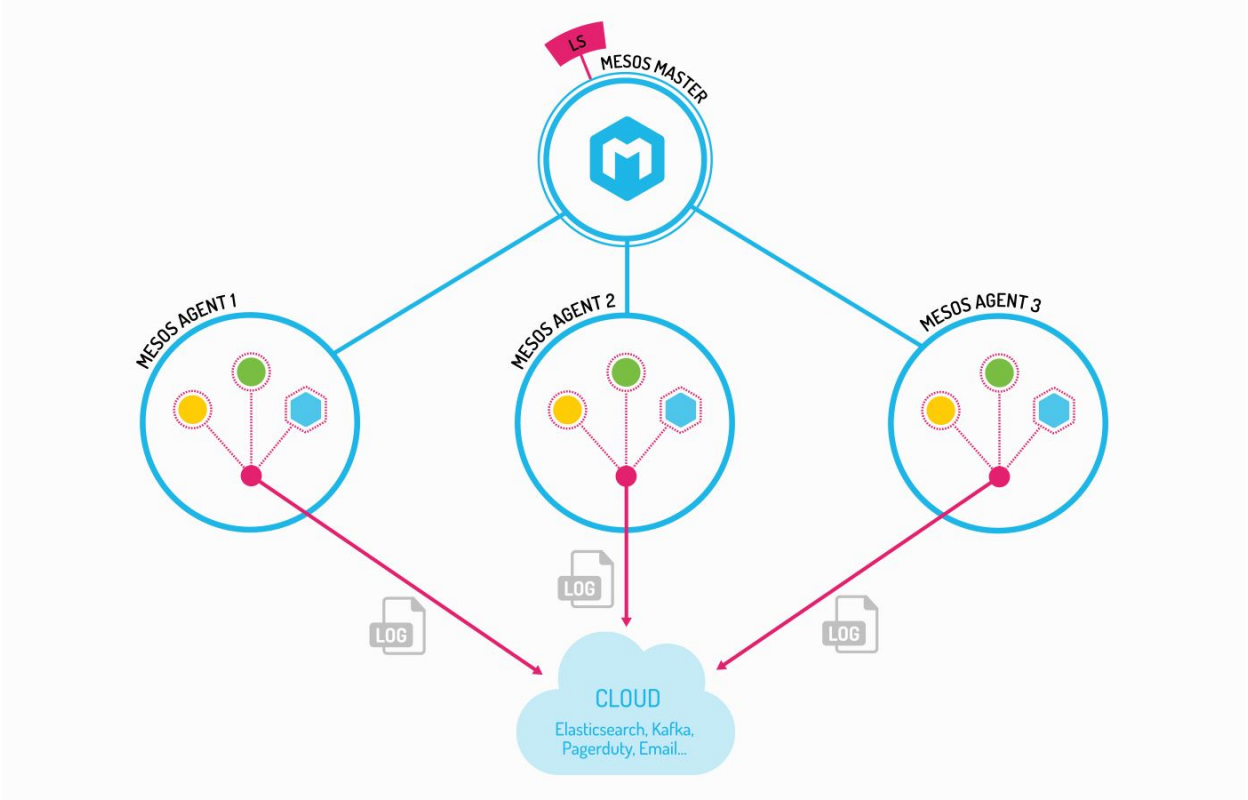
Minimesos - Integrations

1. Configuration file
2. CLI
3. Maven plugin
4. Integration with TravisCI, Jenkins, etc.

Building ELK Mesos Frameworks

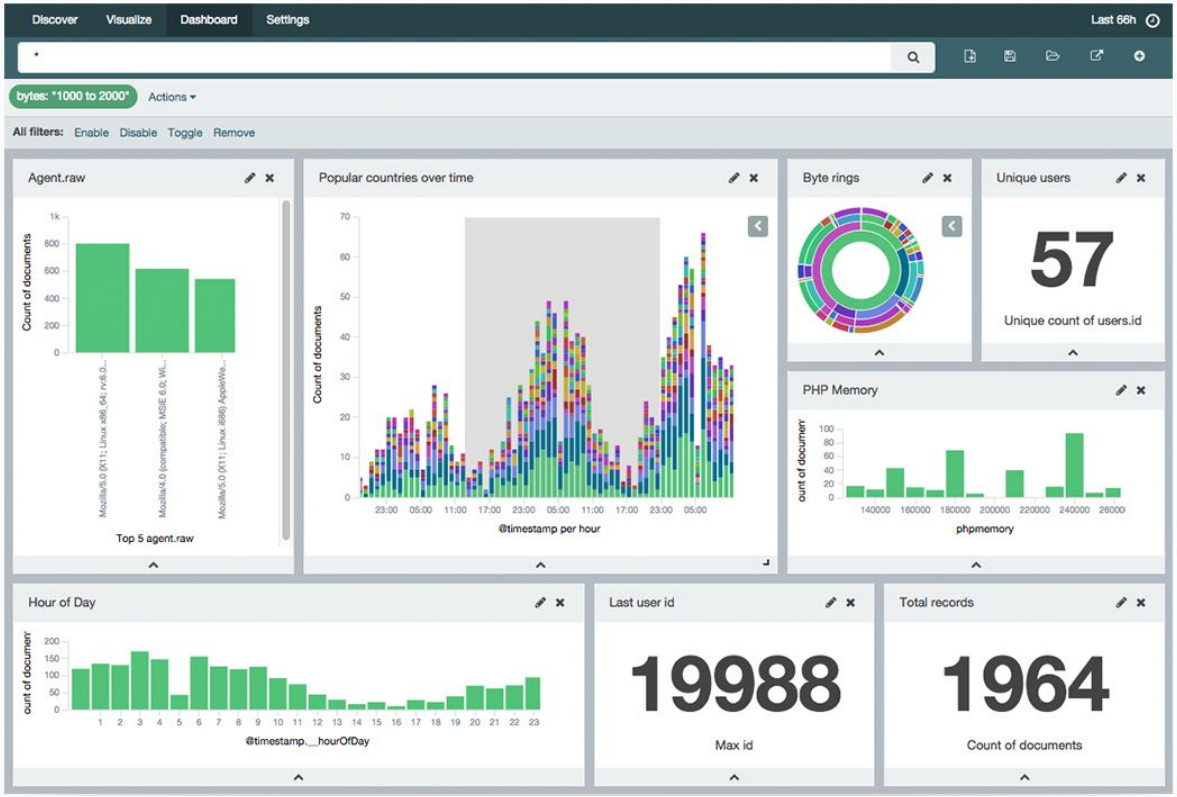


ELK (Logstash)



mesos / logstash

ELK (Kibana)



mesos kibana

Mesos Monkey

Mesos Monkey is a project to test and monitor the applications that are running on your Mesos infrastructure.

It exposes a flexible REST based framework to kill, monitor and stress test your applications

Mesos Monkey

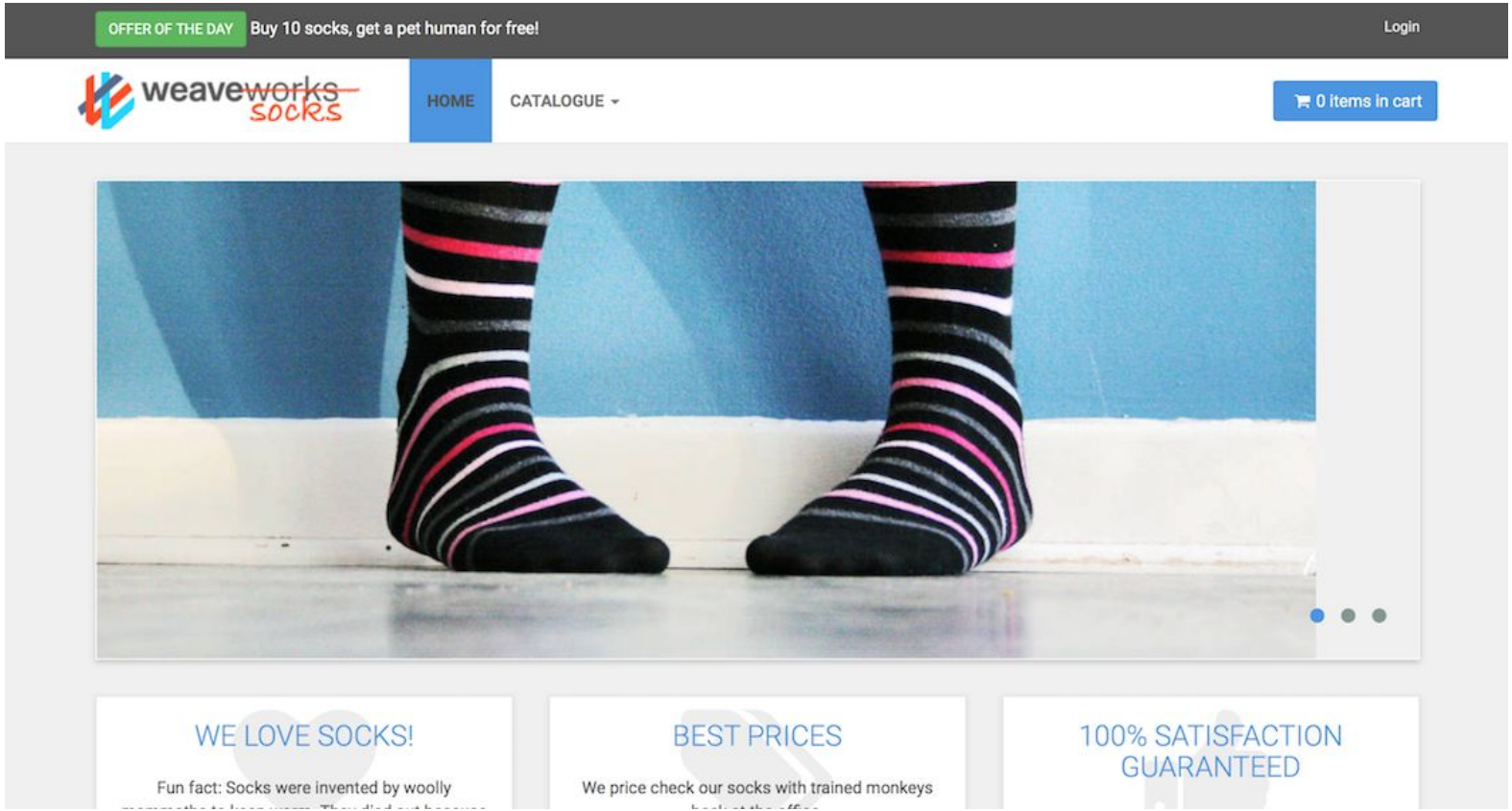
- Kill tasks
- Stress test services
- Monitor performance

- Fully customisable, completely overwritable

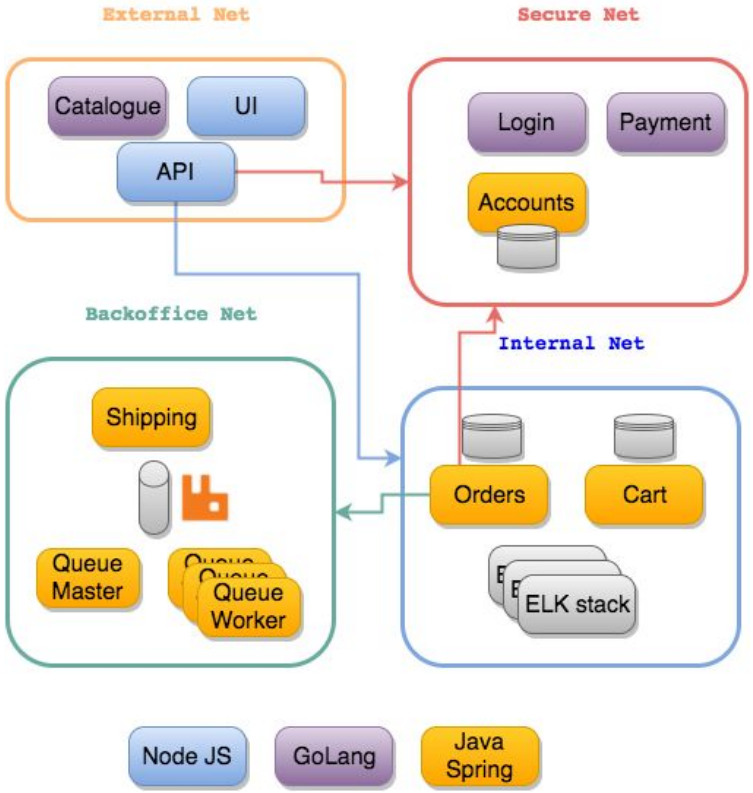
- Start Mesos Monkey
- Write a JSON file to define a Monkey “Job”
- Submit the “job” to an API
- Observe and monitor results

Next big project

Microservices Demo - Weave Socks



Weave Socks Architecture



Weave Socks Containers View

How do we test the Socks Shop?

- Unit
- Component
- Container
- APIs
- Integration
- Advanced

Unit test

- Same as before microservices

Component test

- Test the service without external dependencies. Use data fixtures.

Container test

- Controlled injection of (mocked) dependencies.
- Testing the behaviour under different circumstances.

APIs

- Create API specification to establish a contract between the producer and an API consumer(we use Swagger)
- Validate API specification against the endpoints (we use Dredd)

DREDD

No more outdated API Documentation

Dredd is a language-agnostic command-line tool for validating API description document against backend implementation of the API.

Integration

- Private test environments to allow e2e testing (like minimesos, minukube, docker compose, etc.)
- Collaborative test environments to test combinations and groups of services

Advanced

- Stress, performance and other advanced tests
- Chaos Monkeys
- Services oriented logging, monitoring, tracing and debugging
- A/B testing and Blue/Green or Canary deployments

Summary

Two most important recommendations for testing Cloud Native Apps:

- Build private testing environments
- Test the API producer/consumer contracts

Demo

Questions?