DevOps and Continuous Architecting with TOSCA

Damian A. Tamburri

Technical University Eindhoven and
Jeronimus Academy of Data Science (NL)

My Wheel of Life

B.Sc.

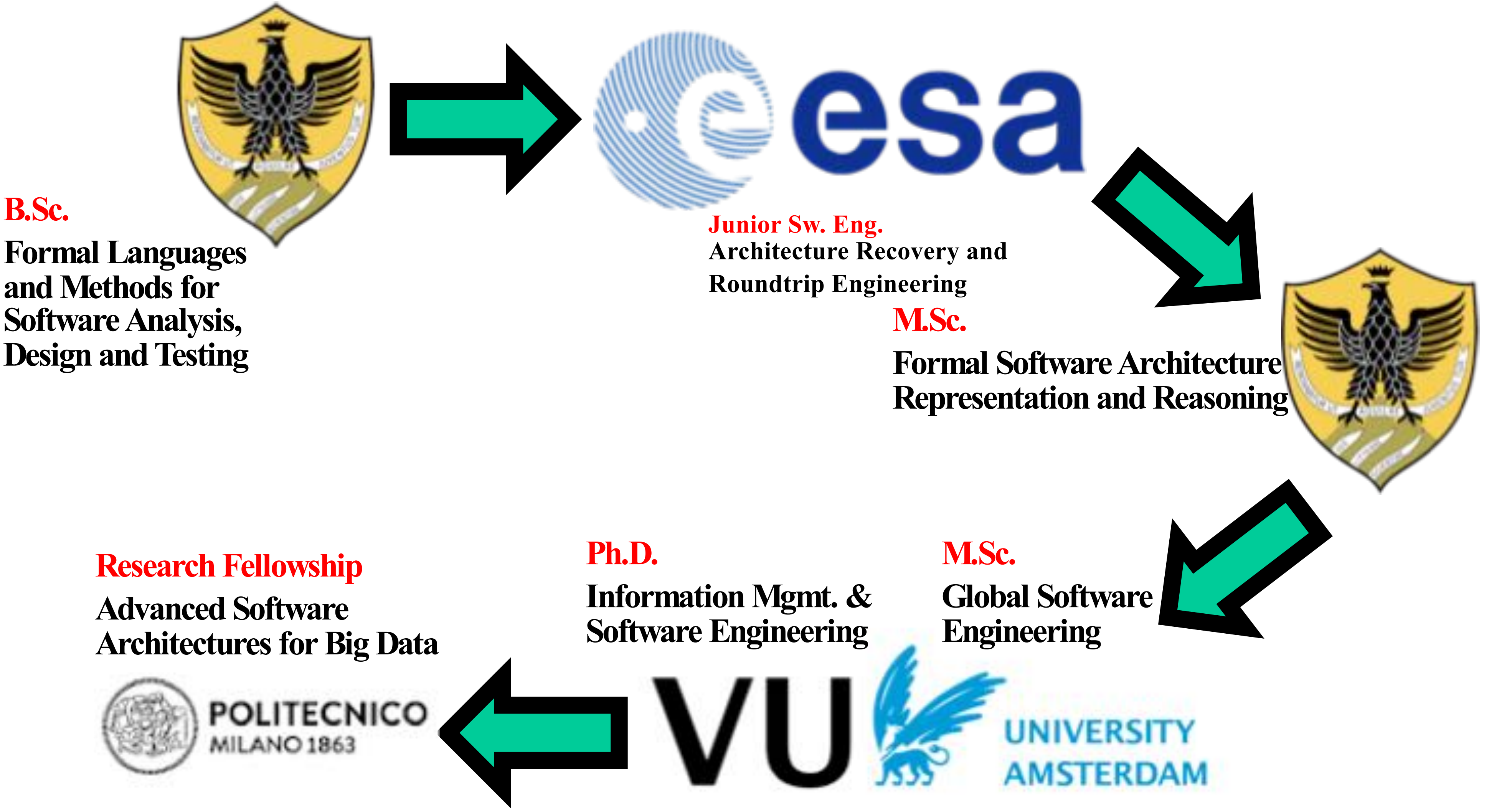
Formal Languages and Methods for Software Analysis, Design, and Testing

Junior Sw. Eng.

Architecture Recovery and Roundtrip Engineering

Mission Accomplished*!

That was a lot of hair ago...



B.Sc.

Formal Languages and Methods for Software Analysis, Design and Testing

Junior Sw. Eng.

Architecture Recovery and Roundtrip Engineering

M.Sc.

Formal Software Architecture Representation and Reasoning

M.Sc.

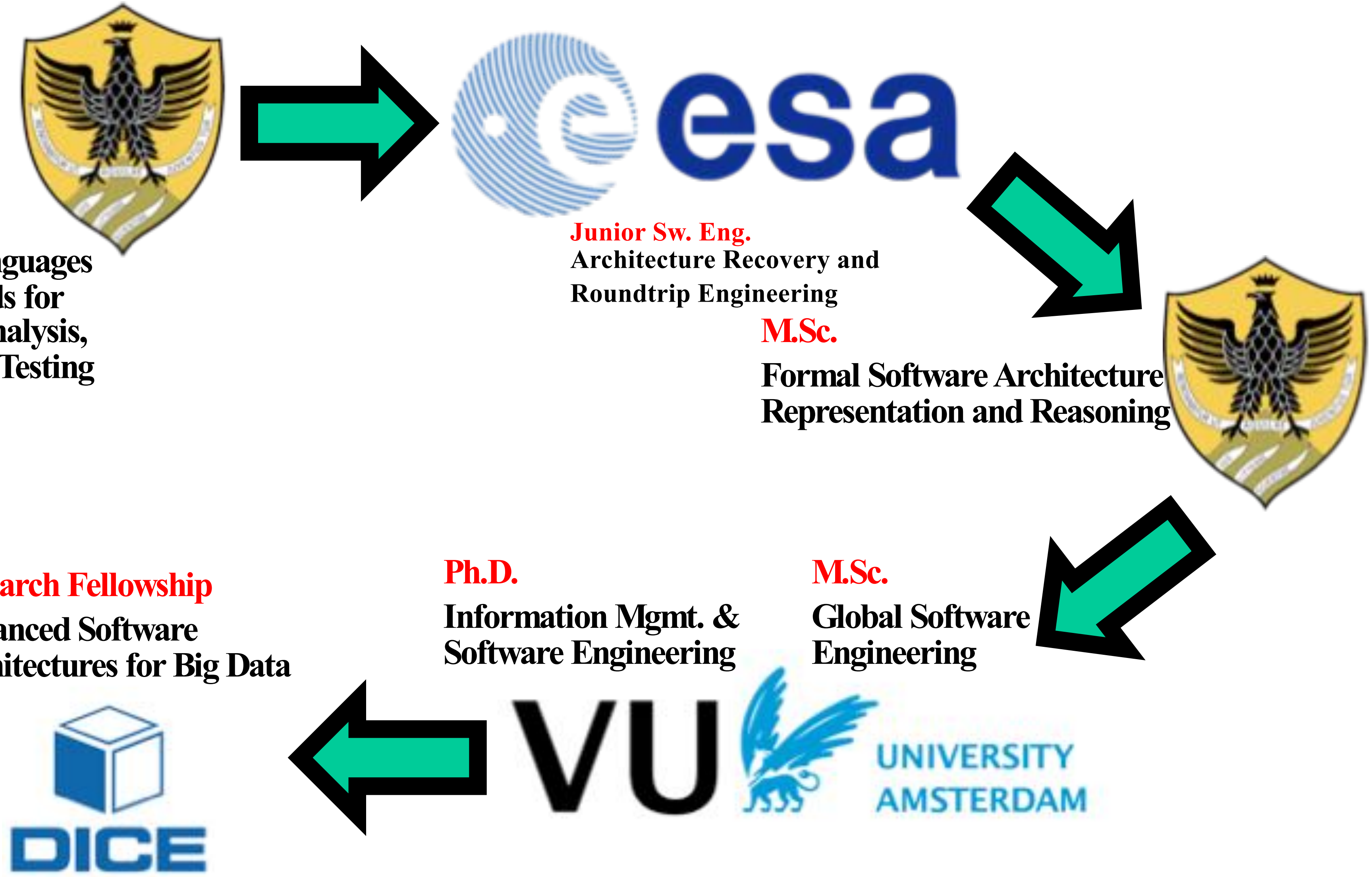
Global Software Engineering

Ph.D.

Information Mgmt. & Software Engineering

Research Fellowship

Advanced Software Architectures for Big Data



B.Sc.

Formal Languages and Methods for Software Analysis, Design and Testing

Junior Sw. Eng.

Architecture Recovery and Roundtrip Engineering

M.Sc.

Formal Software Architecture Representation and Reasoning

M.Sc.

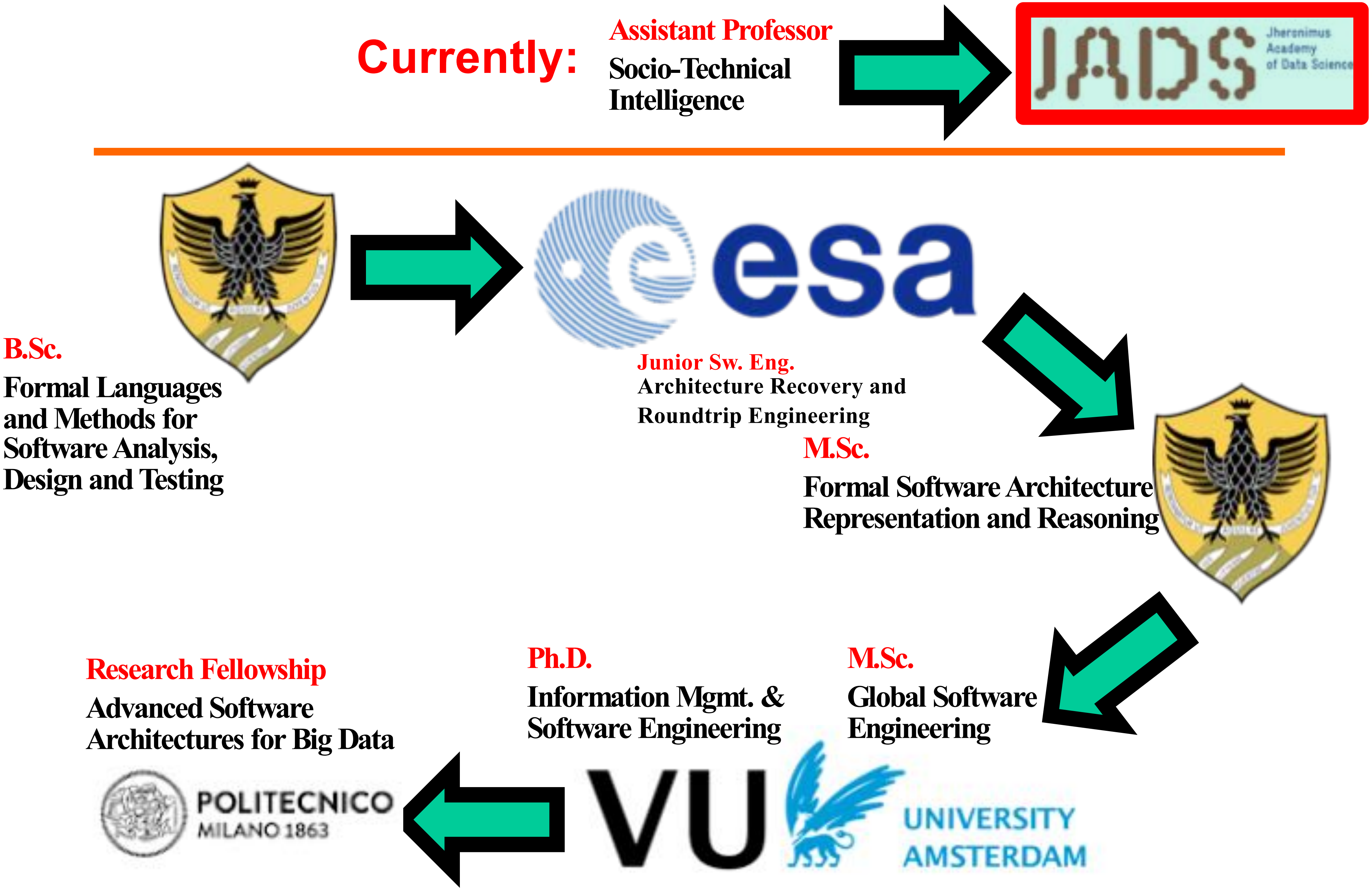
Global Software Engineering

Ph.D.

Information Mgmt. & Software Engineering

Research Fellowship

Advanced Software Architectures for Big Data



Currently:

Assistant Professor

Socio-Technical Intelligence

B.Sc.

Formal Languages and Methods for Software Analysis, Design and Testing

Junior Sw. Eng.

Architecture Recovery and Roundtrip Engineering

M.Sc.

Formal Software Architecture Representation and Reasoning

M.Sc.

Global Software Engineering

Ph.D.

Information Mgmt. & Software Engineering

Research Fellowship

Advanced Software Architectures for Big Data

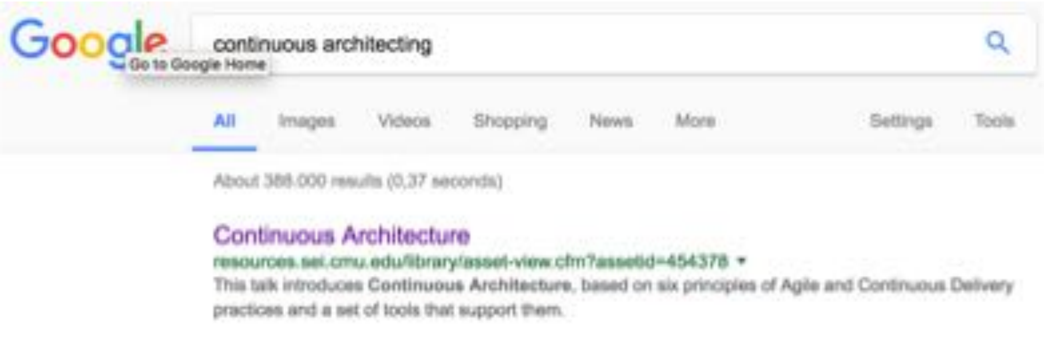
Hot Topic for Today!

- Continuous Architecting!
  1. What is it, where does it come from (i.e., DevOps)
  2. Where does TOSCA fit in
     - Digest: w.t.h. is this TOSCA already???
  3. Continuous Architecting with TOSCA
     - The simple way: orchestrators make arch. Decisions
     - The hard way: orchestrator controls entire process
  4. Continuous Architecting with TOSCA: a real example of the simple way!
  5. Conclusions & Take-home messages

Hot Topic for Today!

- Continuous Architecting!
  - *“Say What????????”*

Not something I, or EU DICE invented...



Google

continuous architecting

About 388.000 results (0,37 seconds)

Continuous Architecture
resources.sei.cmu.edu/library/asset-view.cfm?assetid=454378 ▾
This talk introduces Continuous Architecture, based on six principles of Agile and Continuous Delivery practices and a set of tools that support them.

Continuous Architecture in Practice
Giving an Architecture Perspective to Continuous Delivery

Recent Updates

Role of the Architect

The role of the architect is widely accepted in the world of technology but not clearly defined. This blog post provides a summary of the views we expressed in our book[] as well as in an article published in IEEE Software[].

In general, the term architect exemplifies individuals who make key decisions about the structure and behavior of a system – be that software or other technology areas. Though software architect is the most commonly used term, there are network architects, storage architects, information and security architects. You could even say that any technology subject area has people that identify themselves as architects.

All these types of architects appear to share the following characteristics:

Recent Posts
- Role of the Architect
- Delaying Decisions in Continuous Architecture
- Requirements in Continuous Architecture: Let's Clarify
- Continuous Architecture and Requirements
- Continuous Architecture and Requirements

Continuous architecture in a large distributed agile organization
- a case study at Ericsson

LeanArch.eu
Real world Agile, Scrum, Lean and IT architecture

Continuous Architecting & DevOps

A lot is improved since Agile Manifesto. Still, one of the major complaints from non-Agile community is architecture and long term quality. The usual statement is that Agile people don't care about architecture.

Viktor Grgic

Agile / Scrum coach & trainer, software architecture & development

[illegible]

Let’s check the status of your face...

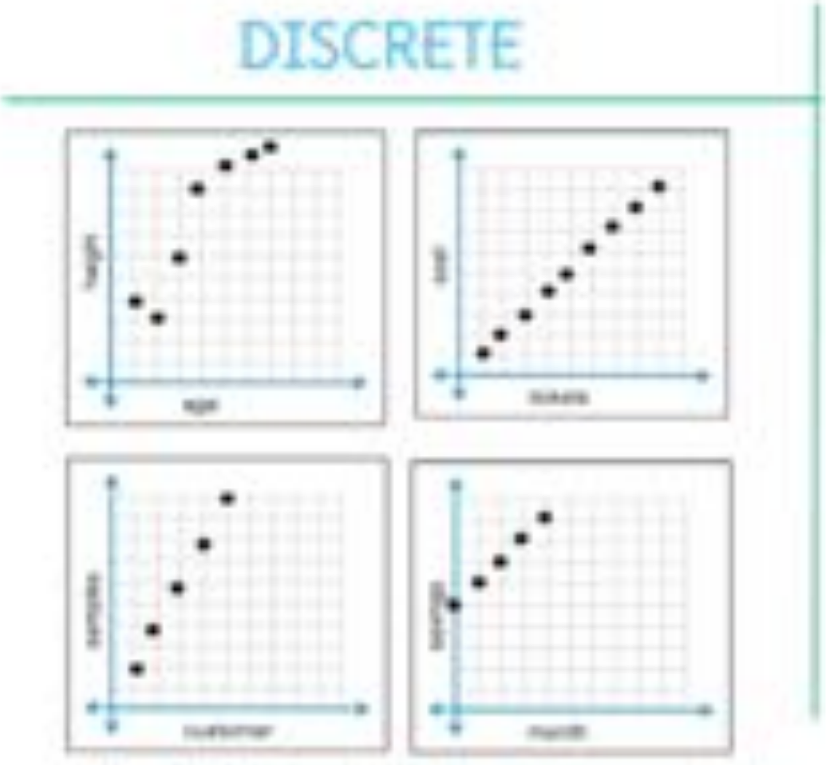
- Continuous Architecting!
  - *“Say What???????”*

My Face when I heard of it...

- Continuous Architecting!
  - *“Say What???????”*

Hot Topic for Today!

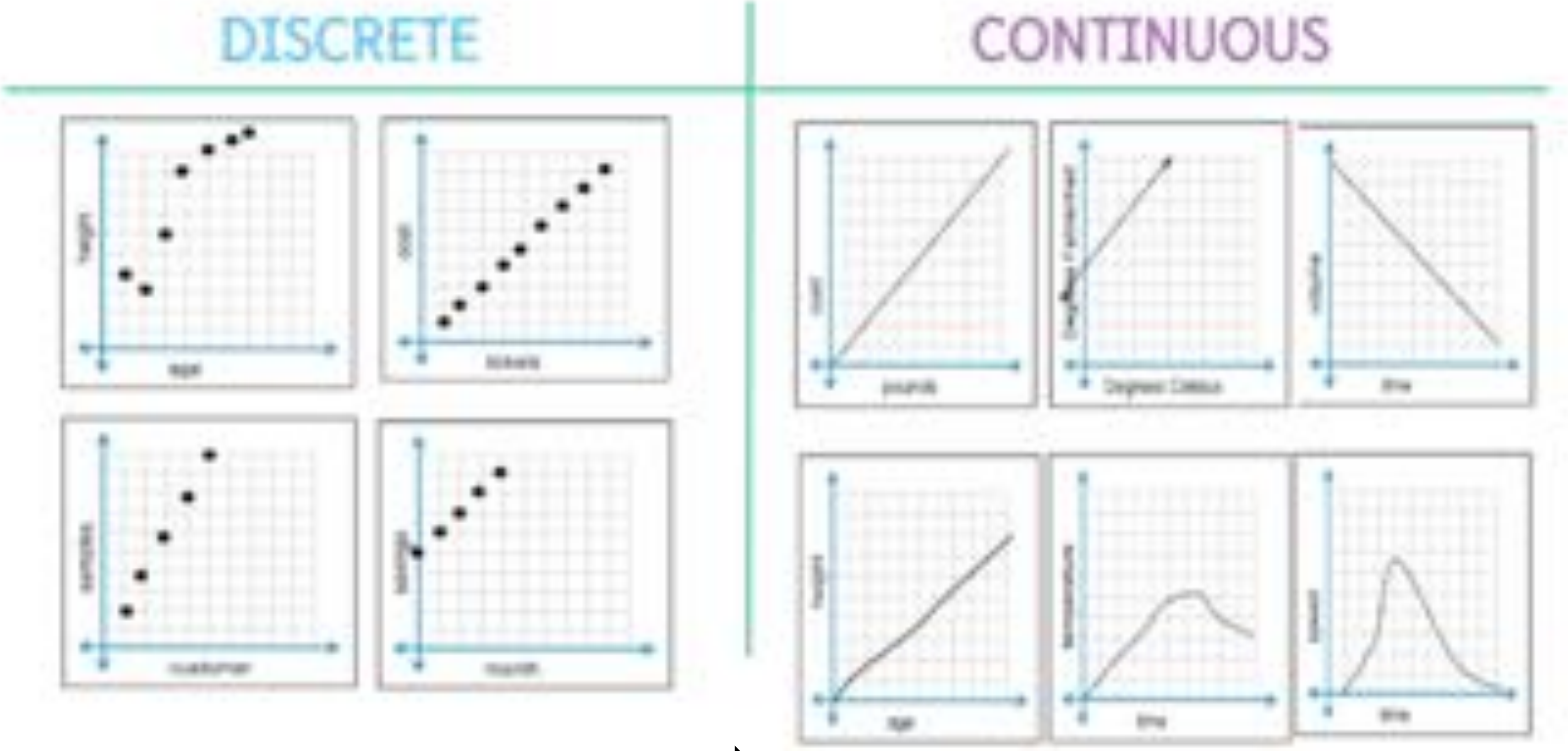
- **Continuous** Architecting!



SAs Before…

Hot Topic for Today!

- **Continuous** Architecting!

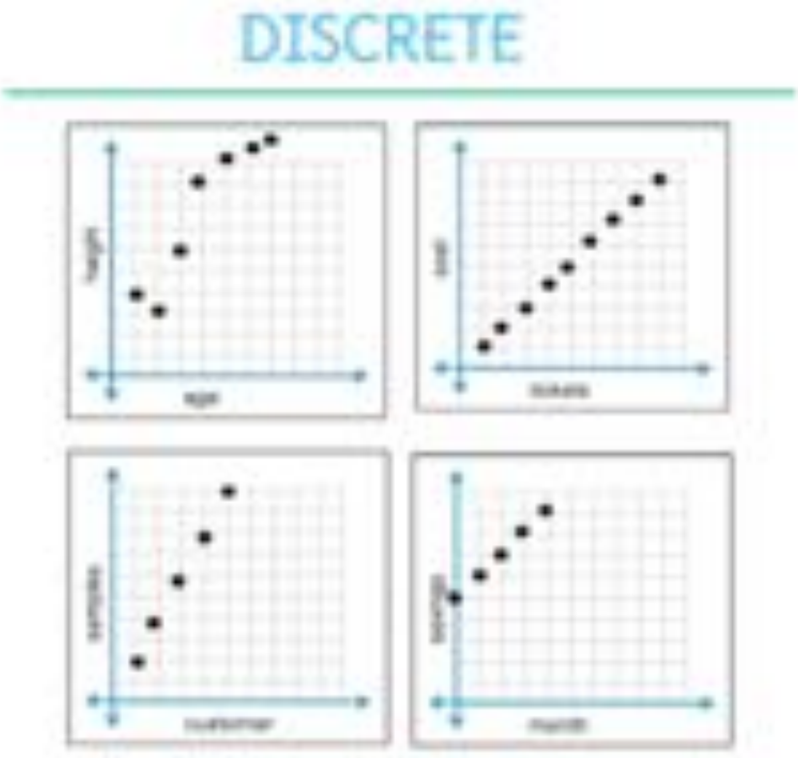


SAs Before…

SAs After!

Hot Topic for Today!

- **Software Architectures**

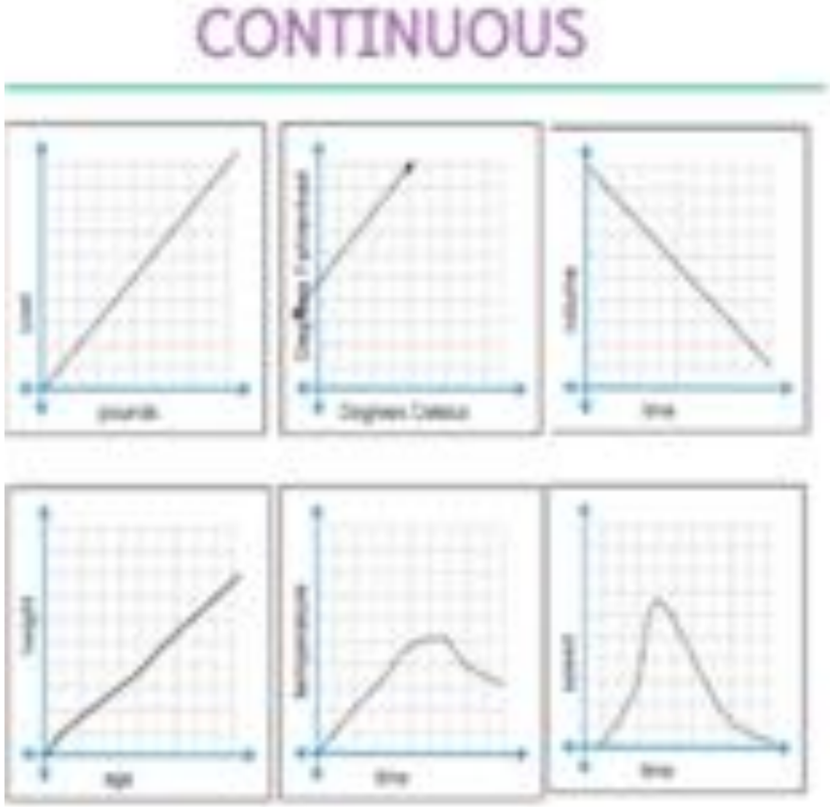


- **Set** of design decisions;
- Assessed before starting implementation, then changed during lifecycle;
- Documented;
- …

SAs Before…

Hot Topic for Today!

- **Continuous** Architecting!
  - Architecture Decisions are not taken, ***they *emerge* in a *data-driven* fashion***;
  - Decision-Making is “Just-in-time”, only where and when ***extremely needed***;
  - Make everything as a **product**, leveraging the ***small*** (Microservices);
  - …



SAs Before…

Hot Topic for Today!

- *“Say What???????”*

My Face when I heard of it...

- Continuous Architecting!
  - *“Say What???????”*

But first... A bit of history!

- Let’s take a step back to where it all began...

It’s 2013…

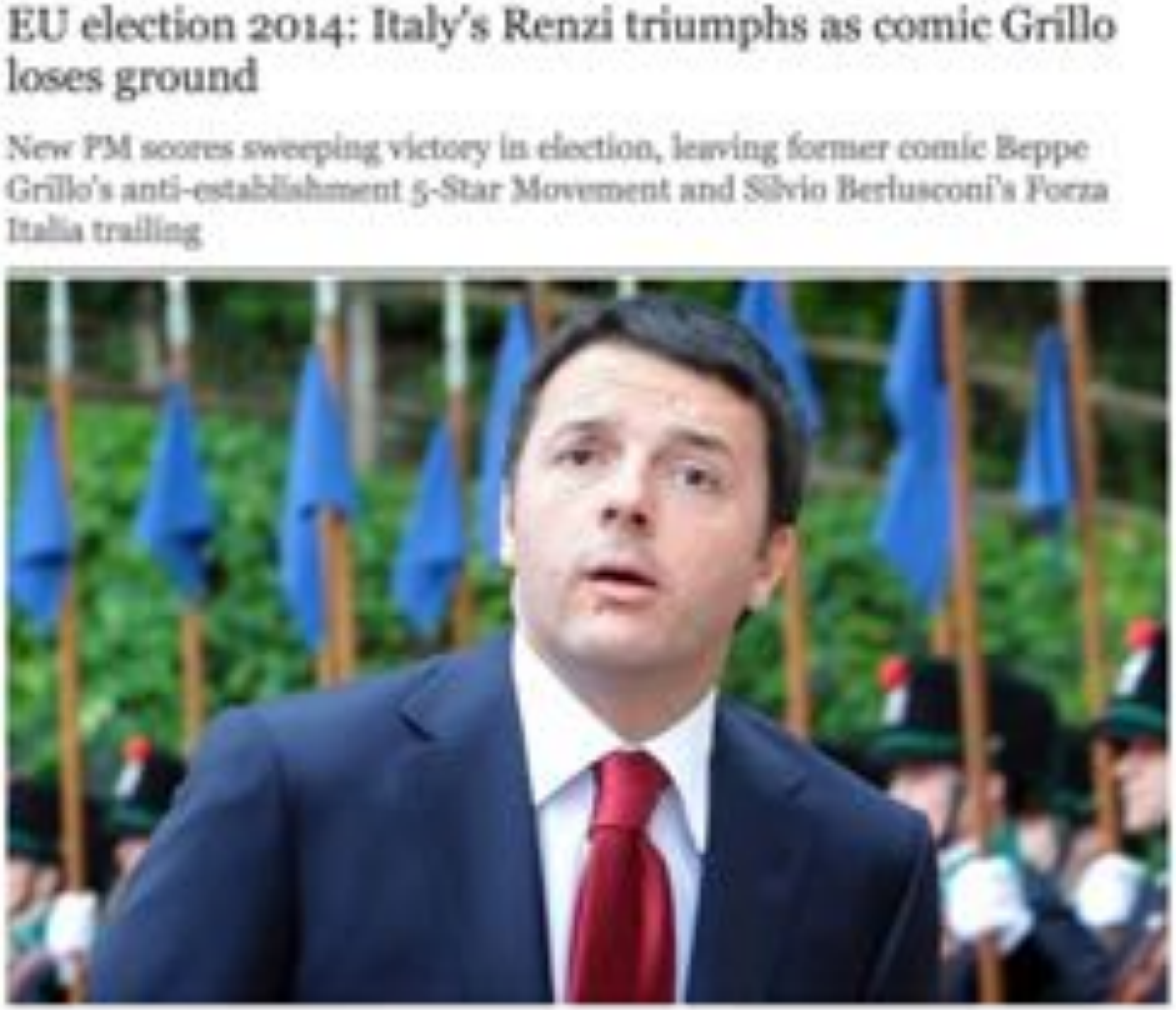
- And…

It’s 2013…

- And…

It’s 2013…

- And…



EU election 2014: Italy's Renzi triumphs as comic Grillo loses ground

New PM scores sweeping victory in election, leaving former comic Beppe Grillo's anti-establishment 5-Star Movement and Silvio Berlusconi's Forza Italia trailing

Meanwhile in Software Engineering... Top failure causes*

- Unrealistic deadlines, e.g., imposed by someone external to the technical staff
- Requirements & people change (too) often
- Effort and resources have been estimated in an overly optimistic way,
- Risks have not been taken into account from the start of the project.
  - Risks can be technical or human difficulties
- Communication problems among staff members
- Difficulty by the management to recognize recurrent delays and take immediate action
- Subversive stakeholders

*Gartner Report 2013

Meanwhile in Software Engineering... Top failure causes* - An Example!

- **Unrealistic deadlines, e.g., imposed by someone external to the technical staff**
- **Requirements & people change (too) often**
- Effort and resources have been estimated in an overly optimistic way,
- Risks have not been taken into account from the start of the project.
  - Risks can be technical or human difficulties
- **Communication problems among staff members**
- Difficulty by the management to recognize recurrent delays and take immediate action
- **Subversive stakeholders**

*Gartner Report 2013

Meanwhile in Software Engineering... Top failure causes* - An Example!

(UNFORESEEN) OVERHEAD COST: 174,000,000 $ (give or take)*

*http://www.cio.com/article/2380827/developer/6-software-development-lessons-from-healthcare-gov-s-failed-launch.html

How is DevOps a response?

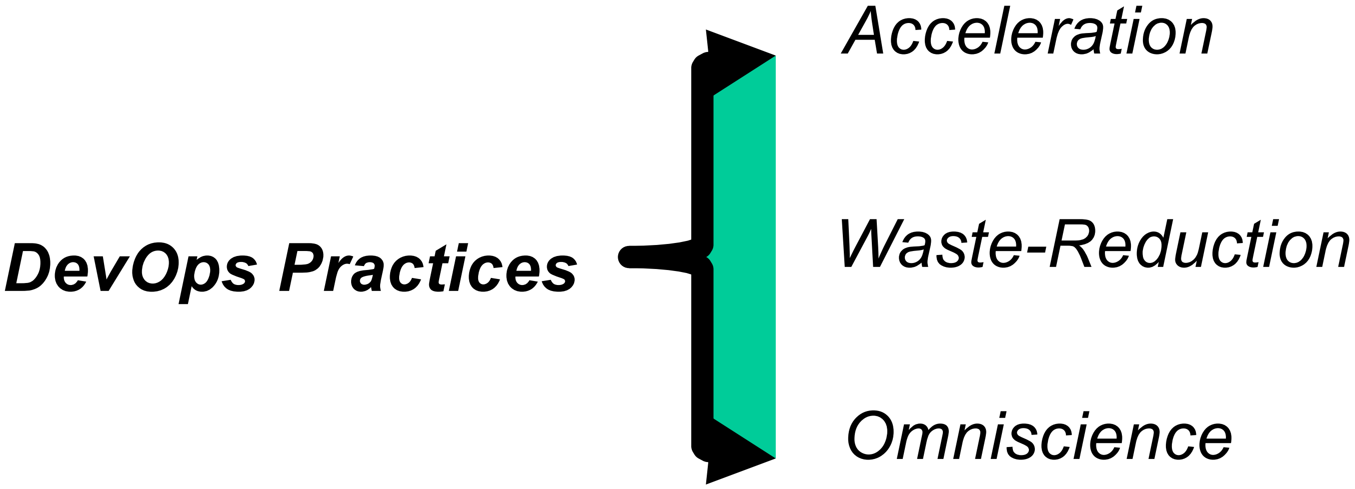
“DevOps is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality.”

L. Bass et Al. [11]

What is DevOps?

*"DevOps is a set of **practices** intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality."*

L. Bass et Al. [11]



DevOps Practices

Acceleration

Waste-Reduction

Omniscience

DevOps Practices: Let’s take a look

- **Acceleration Tactics**
  - Use Faster Organization: Merge Dev+Ops Teams…
  - Infrastructure-as-Code
  - Use Continuous Integration Tools
  - Use Continuous Deployment Tools
  - Use Continuous Testing Tools
  - …
- **Waste Reduction Tactics**
  - Canary Testing
  - A/B Testing
  - Reduce Documentation
  - Minimalistic Architecting ➔ Microservices
  - …

DevOps Practices: Let’s take a look

- **Omniscience Tactics**
  - Monitor Everything
  - Monitoring-as-a-service
  - On-The-Fly Risk Engineering
- …

DevOps Practices: Let’s take a look

- **Omniscience Tactics**
  - Monitor Everything
  - Monitoring-as-a-service
  - On-The-Fly Risk Engineering
  - **Continuous Architecting!**
- ...

Continuous Architecting Explained

- Software Architecture responds to architecture drivers... *So*... “Just” upgrade the drivers for DevOps!
  - Design for Modifiability
  - Design for Observability
  - Design for Organisability
  - Design for Fast Evolution & Testability
  - Design for High Scalability

but… most of all…

Continuous Architecting Explained

- Software Architecture responds to architecture drivers... *So*... "Just" upgrade the drivers for DevOps!

  - Design for Modifiability
  - Design for Observability
  - Design for Organisability
  - Design for Fast Evolution & Testability
  - Design for High Scalability

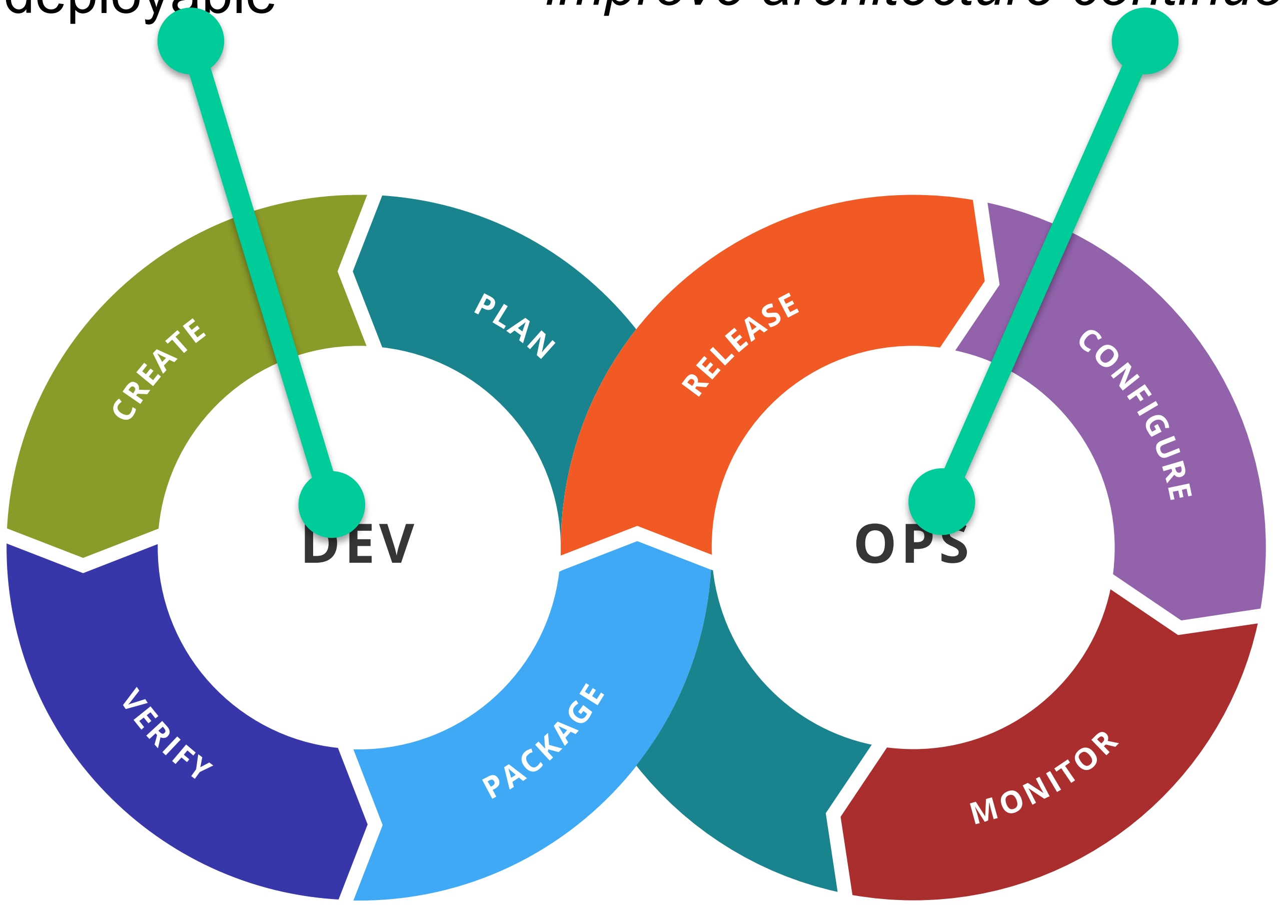
but… most of all…

  - **Design for SA failure!**
    - SA is incremental, refined from a rough draft **via neverending continuous architectural improvement!**

Continuous Architecting In Context

Dev Goal: "Prepare a Software Architecture designed to be immediately deployable"

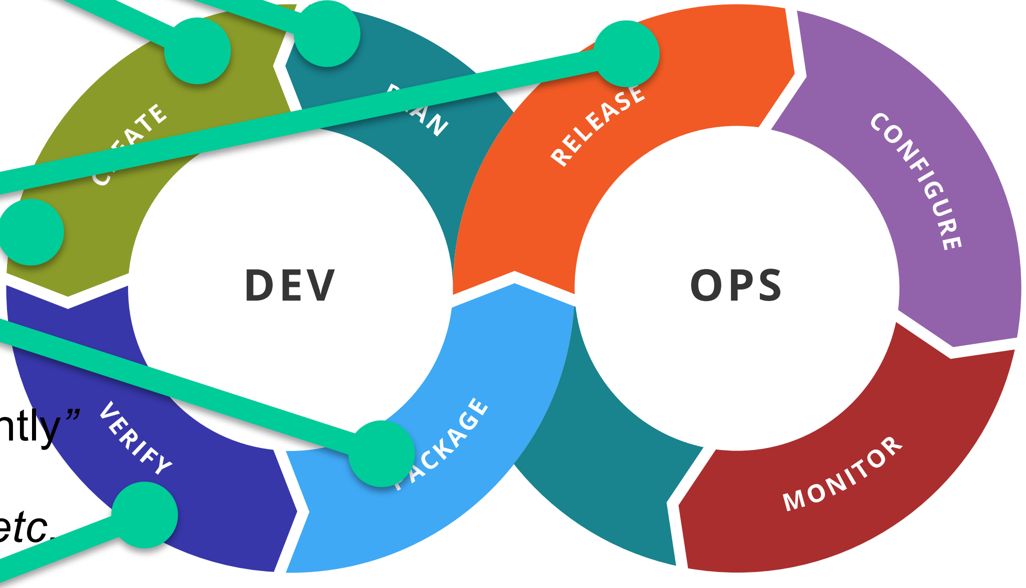
Ops Goal: "Observe the architecture runtime and provide Ops feedback to Dev…*then improve architecture continuously*"



DevOps processes and toolchain: Putting it all together…

- Continuous Architecting

Def. *“architect for test, build and deploy, take quality attributes into account, take advantage of feedback from runtime” [1]*



- Continuous Integration

Def. *“merge* all developer work-copies to a shared mainline frequently” *[4]*

***Examples. Apache Jenkins**, **Hudson**, etc.*

- Continuous Testing

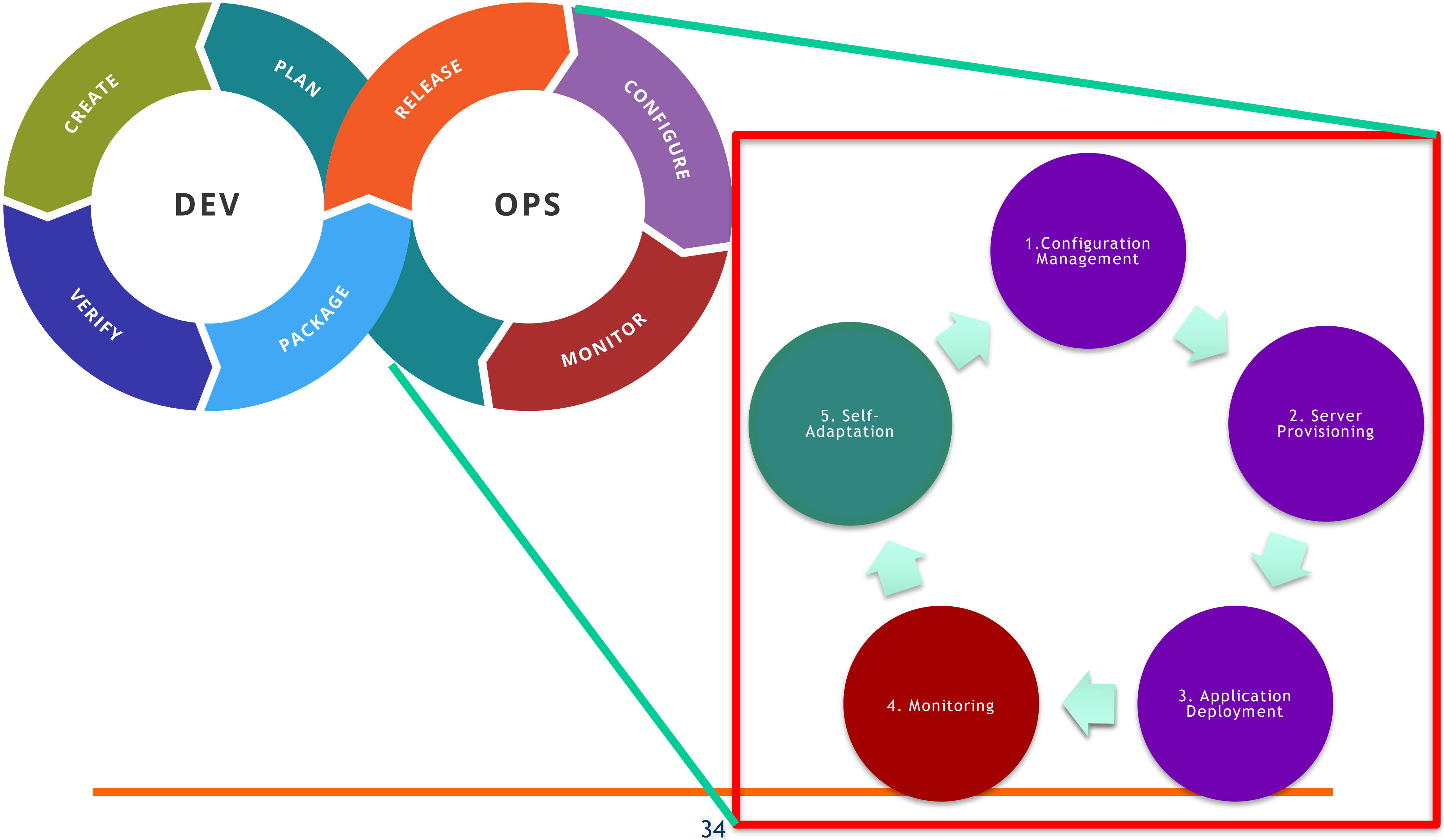
Def. *“run tests as part the build pipeline so that every check-in and deployment is validated” [3]*

***Examples. Selenium+GitHub+LI-API**, etc.*

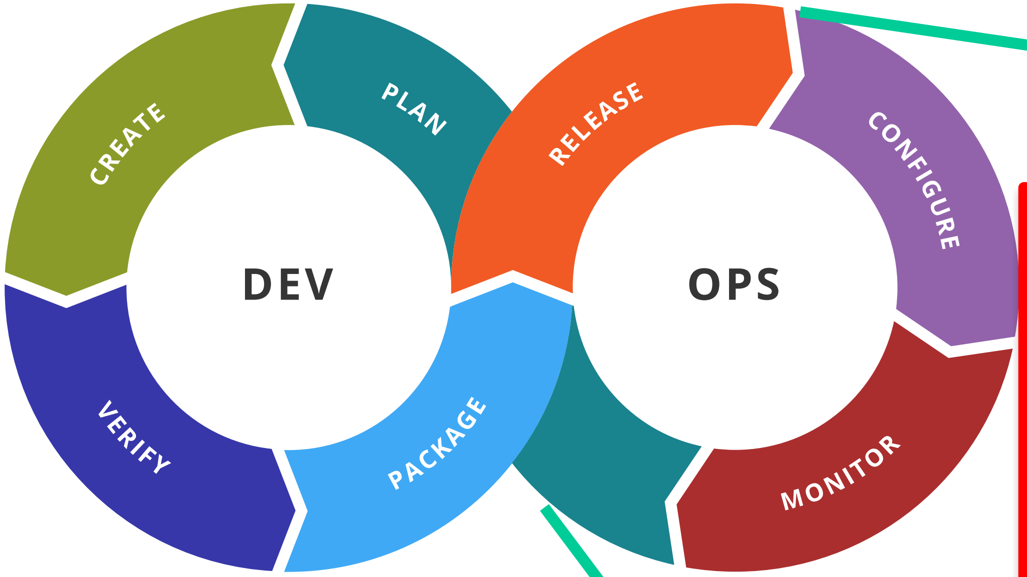
…

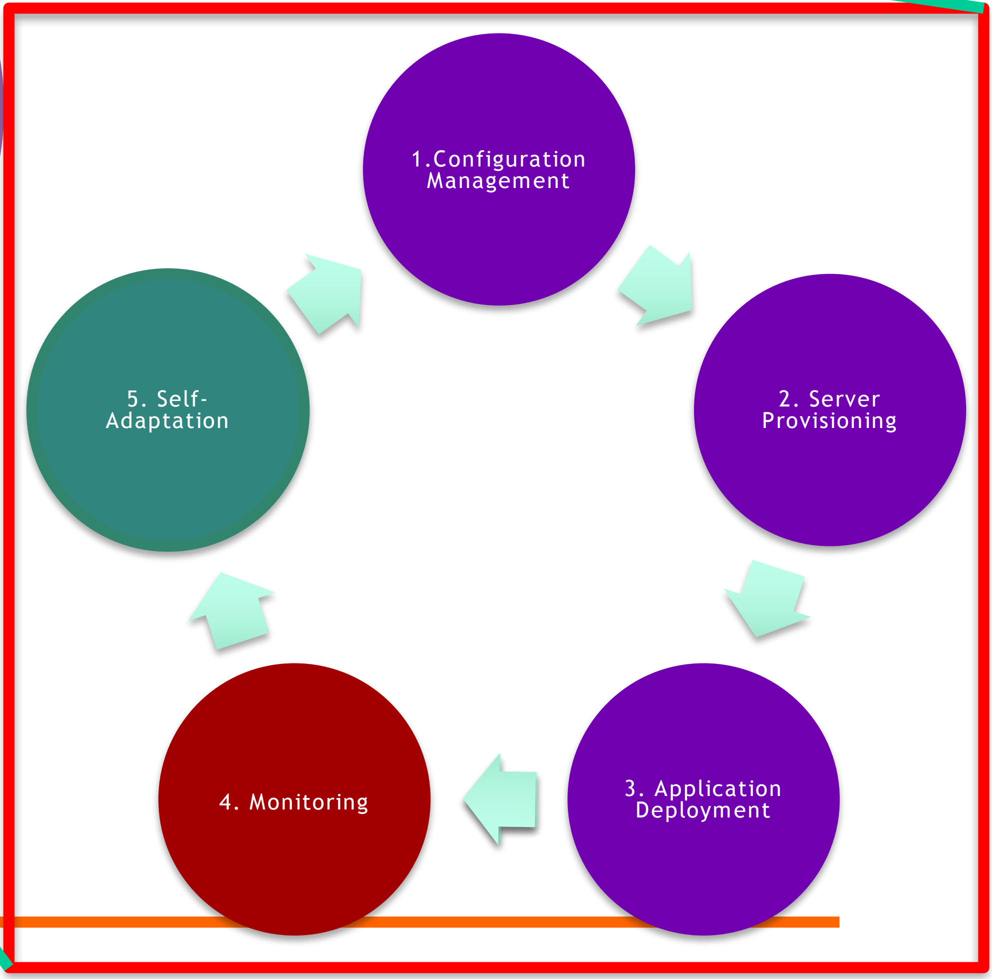
Image by Kharnagy (Own work) [CC BY-SA 4.0 (http://creativecommons.org/licenses/by-sa/4.0)], via Wikimedia Commons

DevOps process and toolchain



DevOps process and toolchain





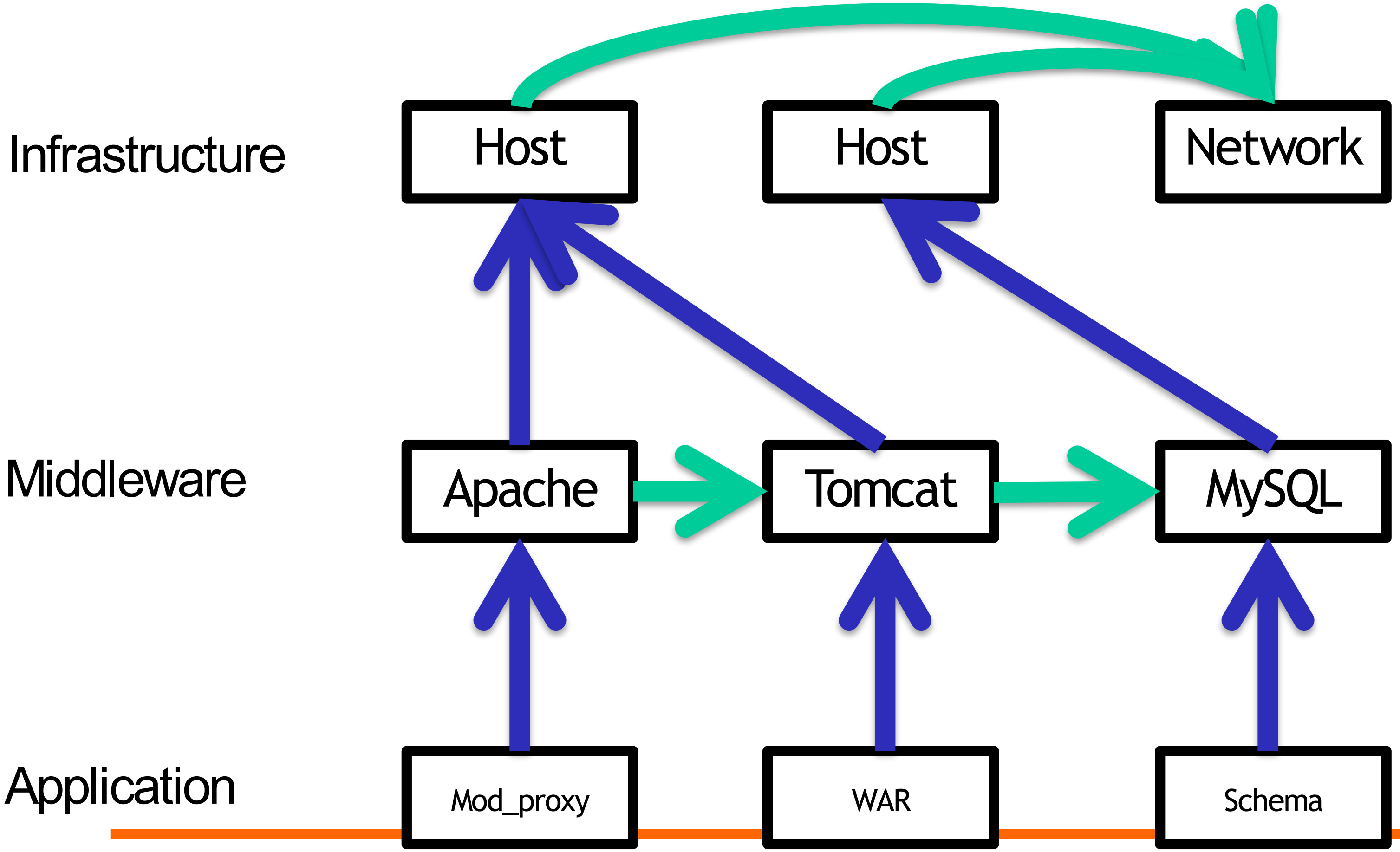
Spoiler-alert!
We're moving towards TOSCA ☺

Continuous Architecting…
Where does TOSCA fit in?

- I'm assuming some of you know TOSCA, but just in case… DIGEST!

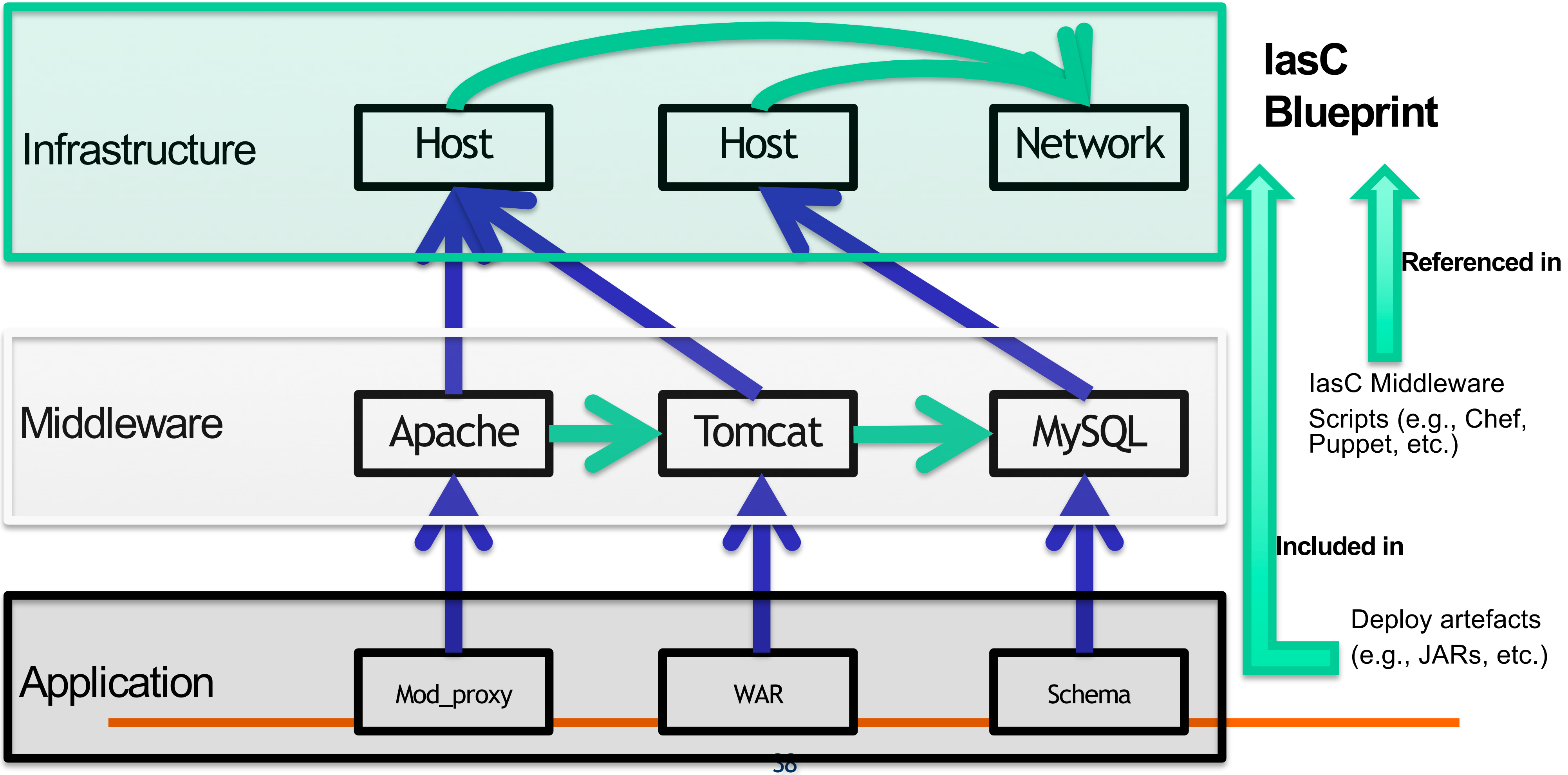
Towards standard Infrastructure Code

➔ **An Application Deployment Topology**, i.e., “a graph of physical artefacts that need support for several lifecycle phases (e.g., procurement, installation, configuration, deployment, undeployment, teardown, etc.)” [6]



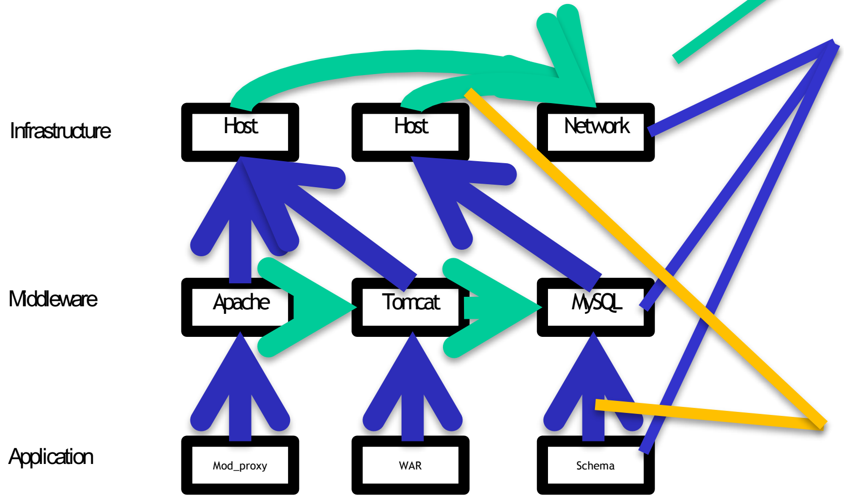
Towards standard Infrastructure Code

➔ **Infrastructure-as-code**, i.e., “a blueprint detailing physical artefacts, all scripts for all lifecycle phases and all artefacts needed for deployment” [6]



Where Does TOSCA fit into?

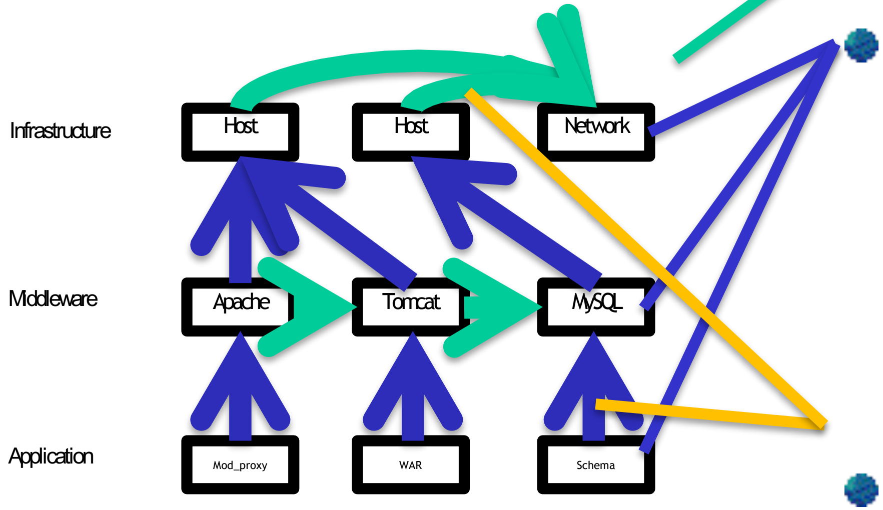
Here’s What We’ve Seen there…



- An application topology
- 3 layers
  - Infrastructure (Cloud or DC objects)
  - Platform or Middleware (App containers)
  - Application modules, schemas and configurations
- Relationships between components:
  - What’s hosted on what or installed on what
  - What’s connected to what

Where Does TOSCA fit into?

Here's What We've Seen there…

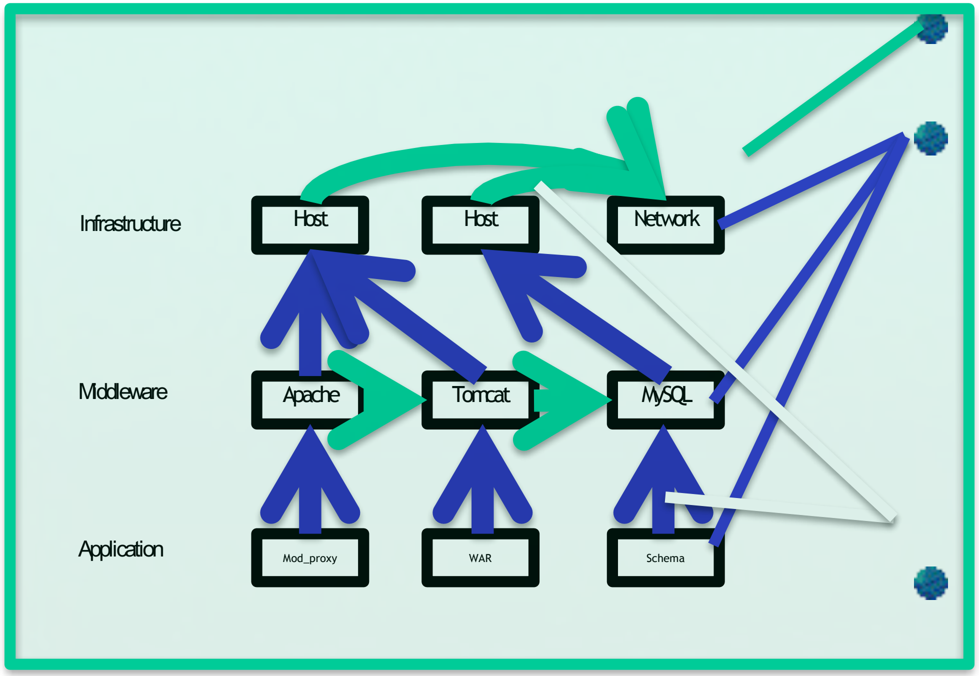


- An application topology
- 3 layers
  - Infrastructure (Cloud or DC objects)
  - Platform or Middleware (App containers)
  - Application modules, schemas and configurations
- Relationships between components:
  - What's hosted on what or installed on what
  - What's connected to what

TOSCA: "**T**opology and **O**rchestration **S**pecification for **C**loud **A**pplications"

Where Does TOSCA fit into?

Here’s What We’ve Seen there…



TOSCA: “**Topology** and **O**rchestration **S**pecification for **C**loud **A**pplications”

An application topology

3 layers

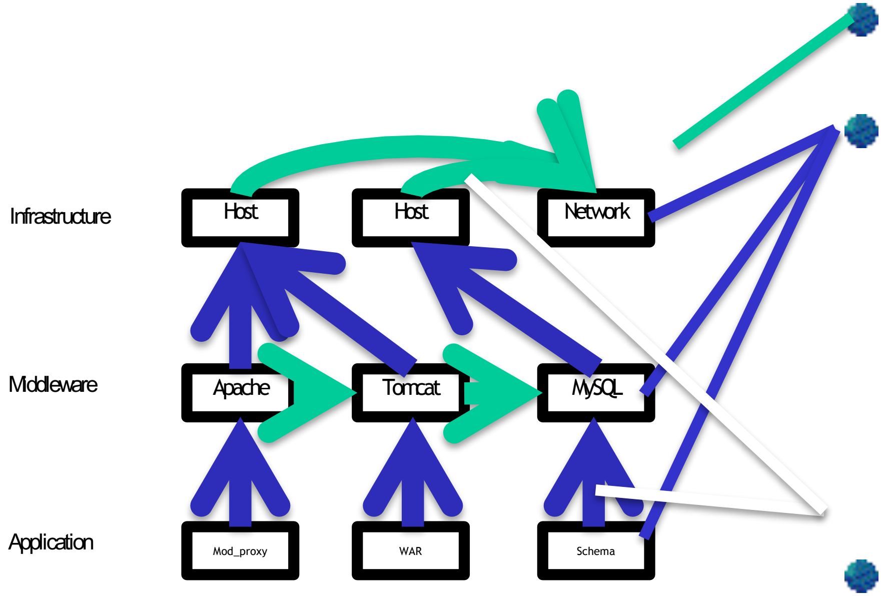
- Infrastructure (Cloud or DC objects)
- Platform or Middleware (App containers)
- Application modules, schemas and configurations

Relationships between components:

- What’s hosted on what or installed on what
- What’s connected to what

Where Does TOSCA fit into?

Here’s What We’ve Seen there…

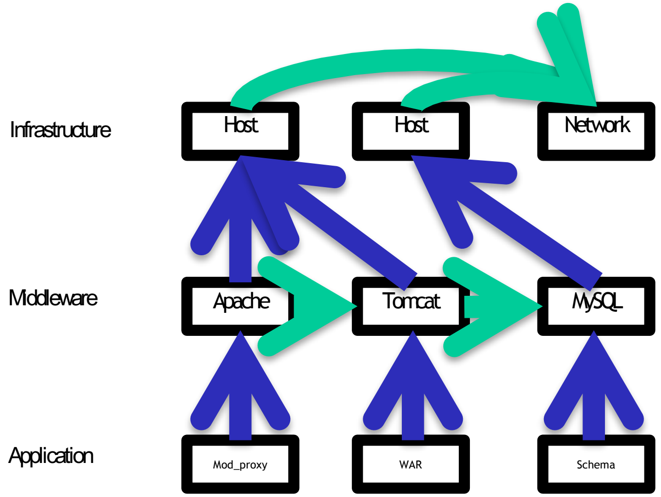


TOSCA: “**T**opology and **Orchestration** **S**pecification for **C**loud **A**pplications”

- An application topology
- 3 layers
  - Infrastructure (Cloud or DC objects)
  - Platform or Middleware (App containers)
  - Application modules, schemas and configurations
- Relationships between components:
  - What’s hosted on what or installed on what
  - What’s connected to what

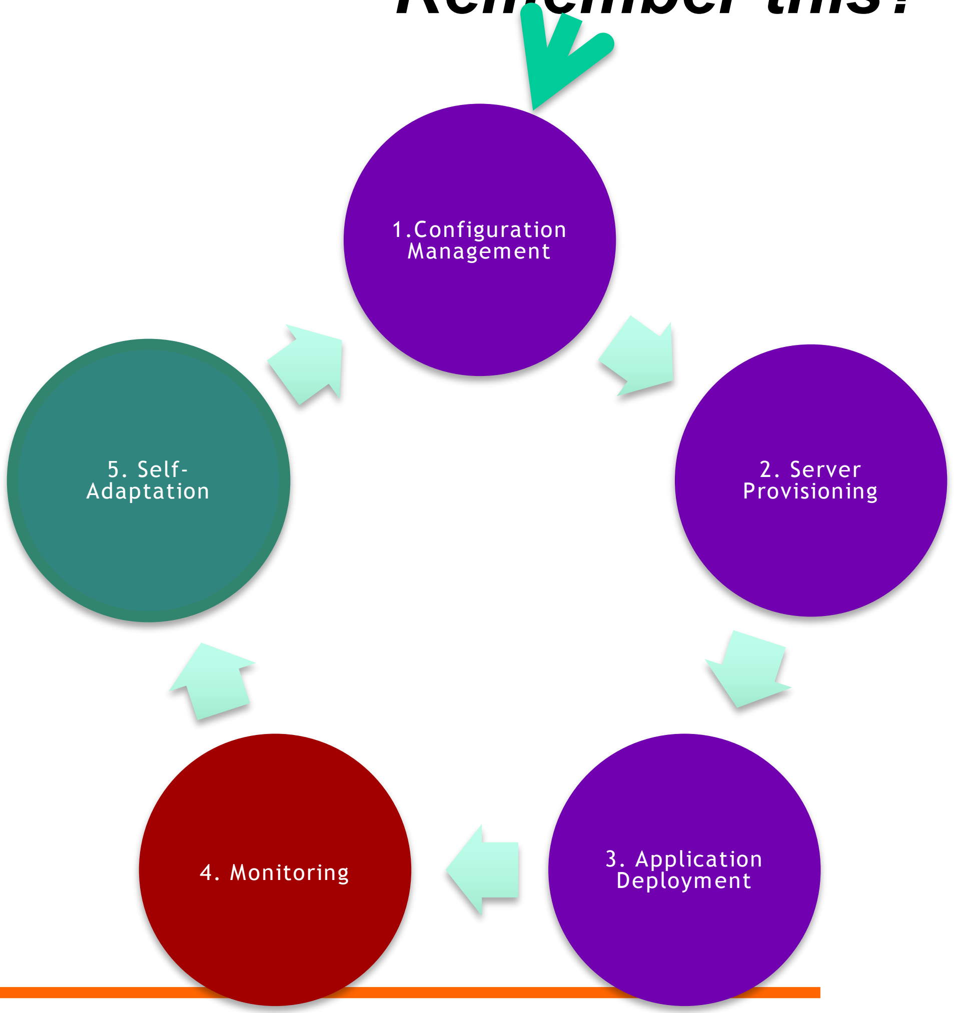
Where Does TOSCA fit into?

Here's What We've Seen there…



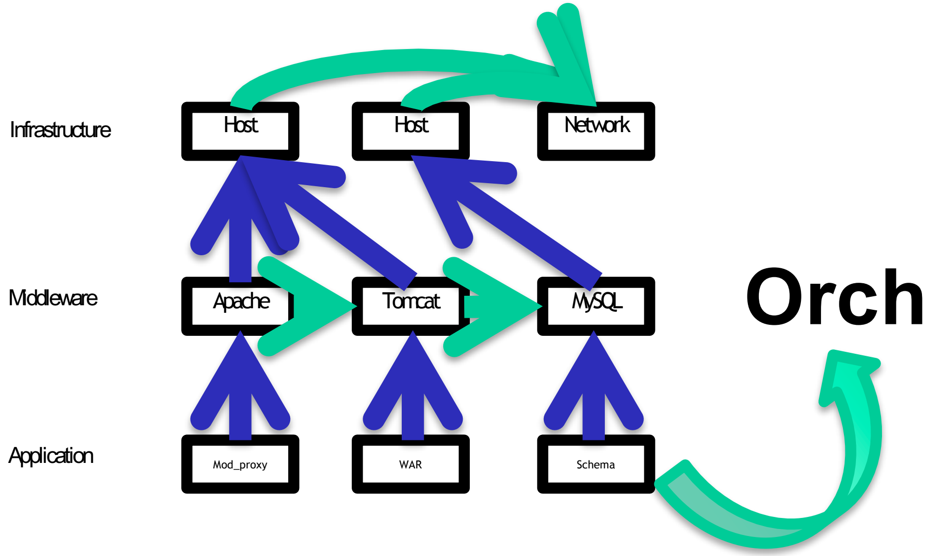
TOSCA: "**T**opology and **Orchestration** **S**pecification for **C**loud **A**pplications"

Remember this?

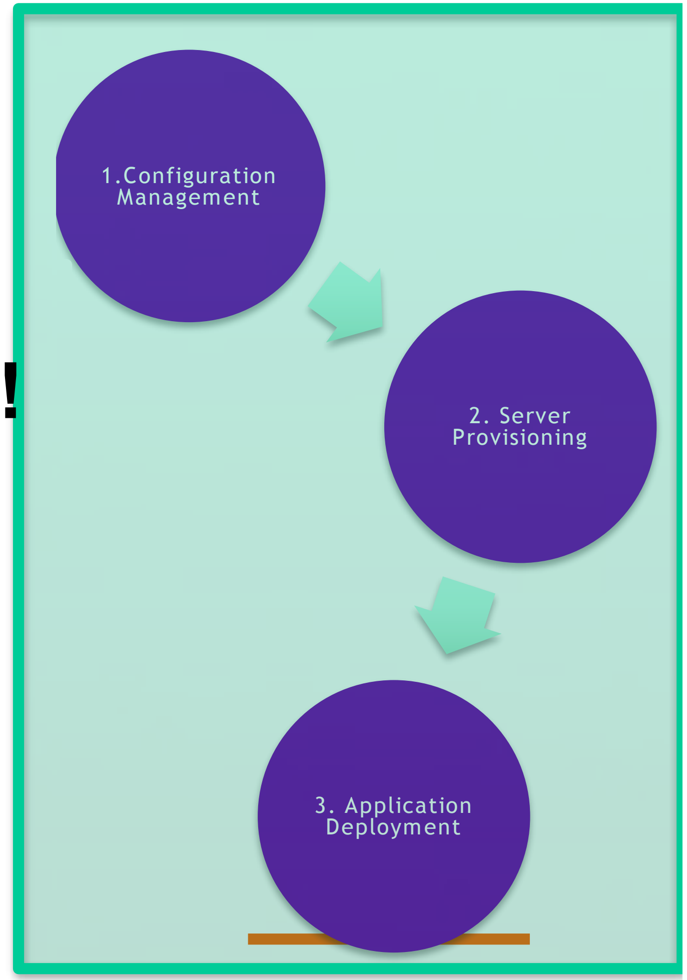


Where Does TOSCA fit into?

Here’s What We’ve Seen there…

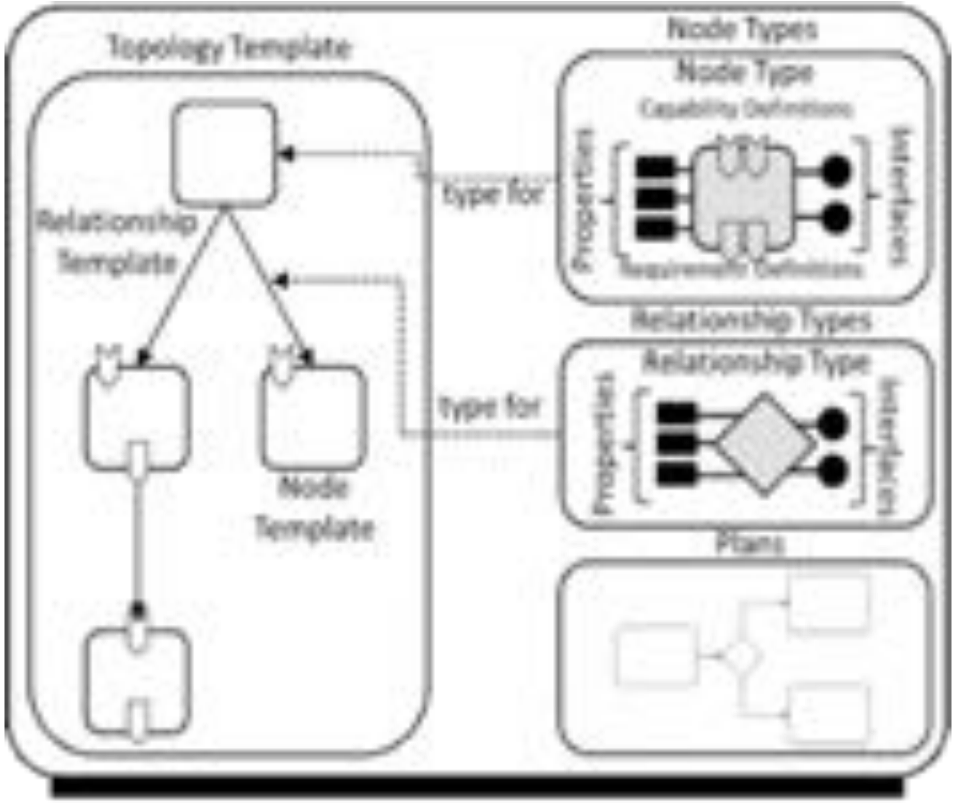


Orchestration!



TOSCA: “Topology and **Orchestration** Specification for Cloud Applications”

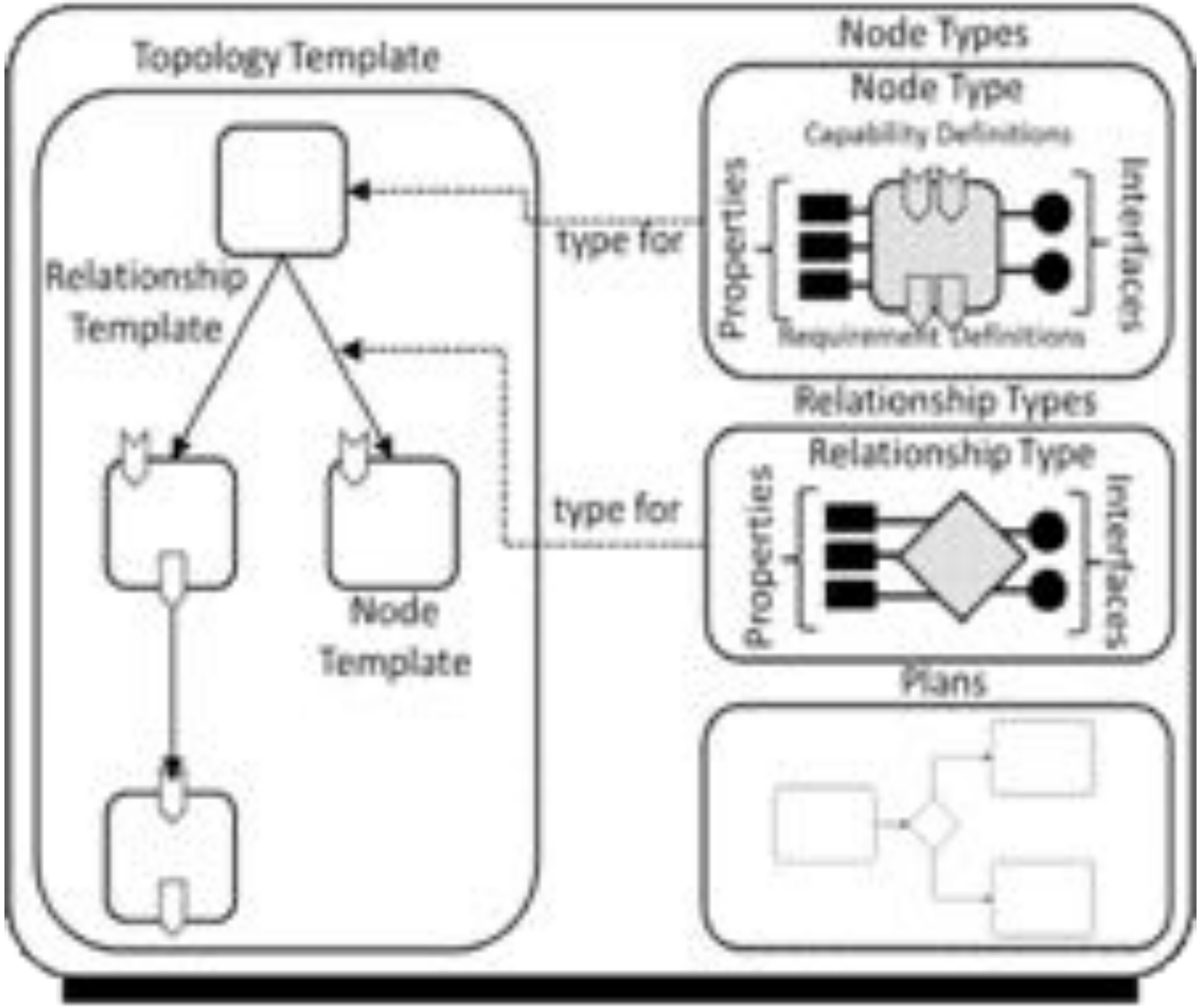
What’s in a TOSCA Topology?



- component in the topology are called Nodes
- Each Node has a Type (e.g. Host, BD, Web server).
  - The Type is abstract and hence portable
  - The Type defines Properties and Interfaces
- An Interface is a set of hooks (named Operations)
- Nodes are connected to one another using Relationships

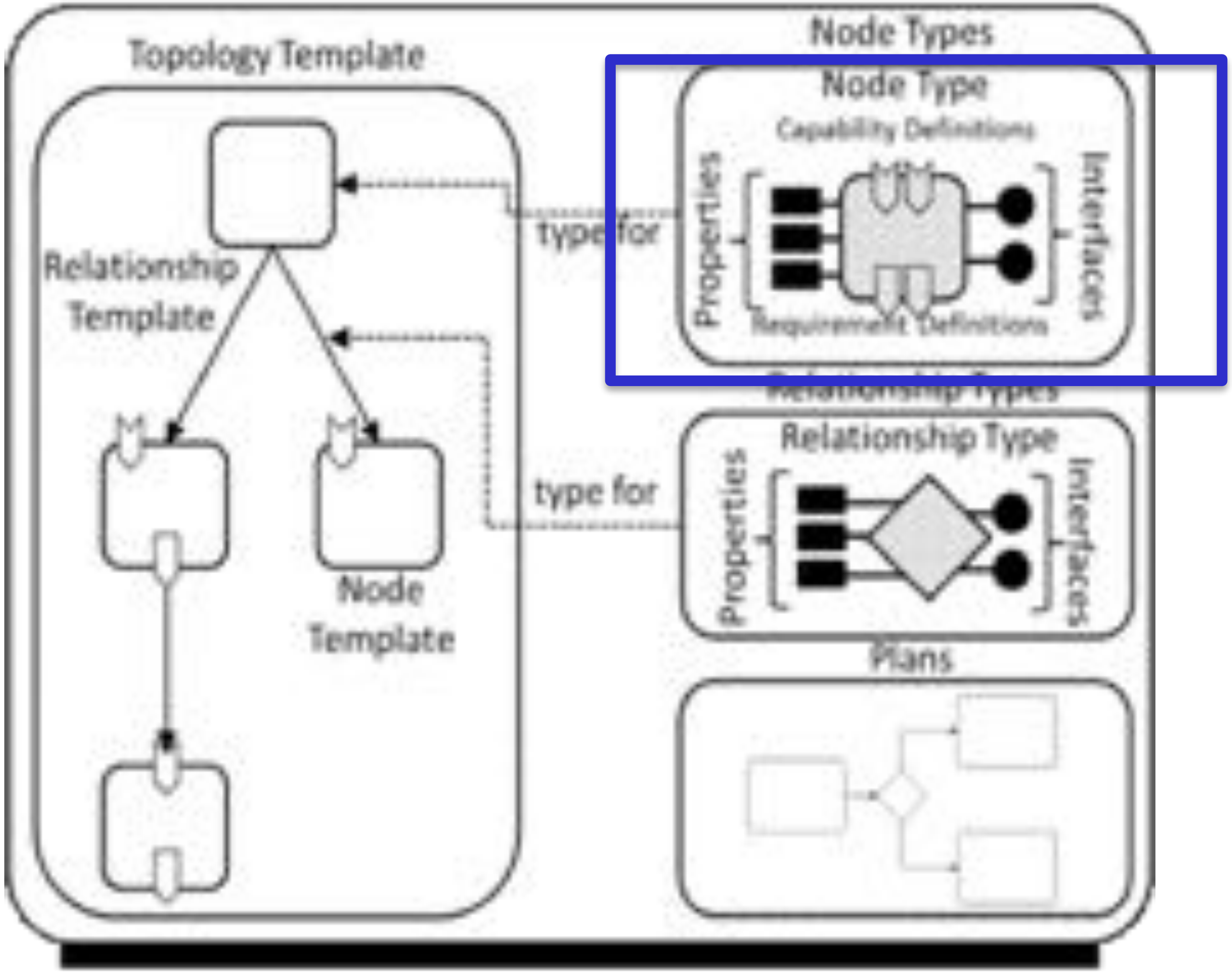
TOSCA, IasC Standard

TOSCA Service Template [7]



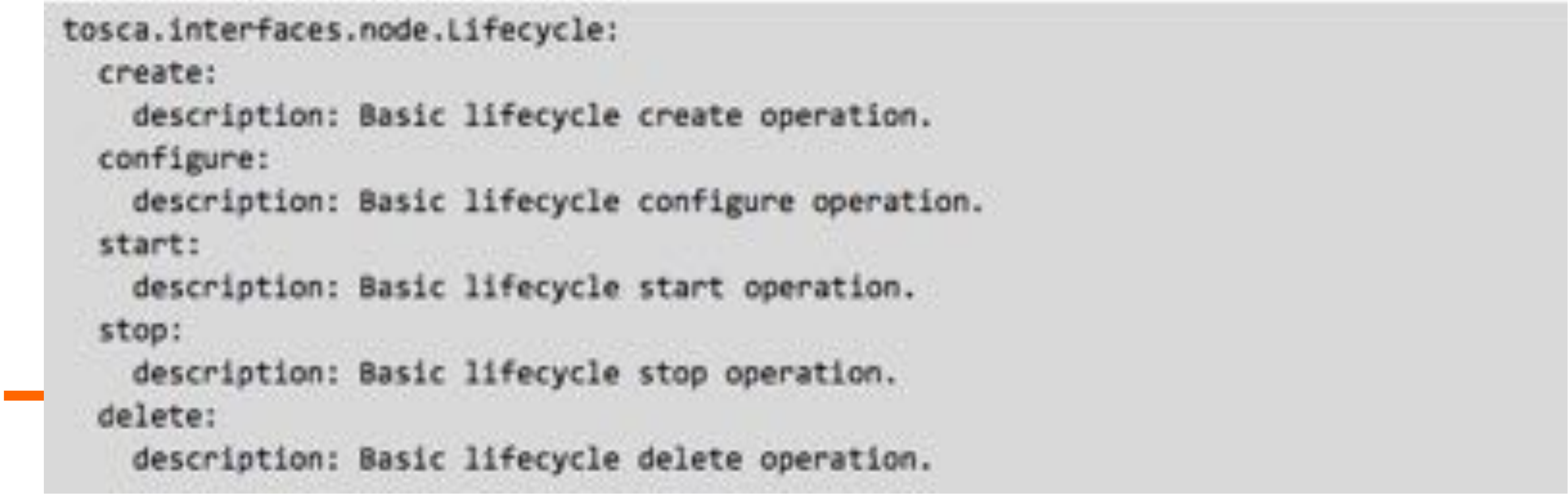
TOSCA Core Ingredients

TOSCA Service Template



Node Type

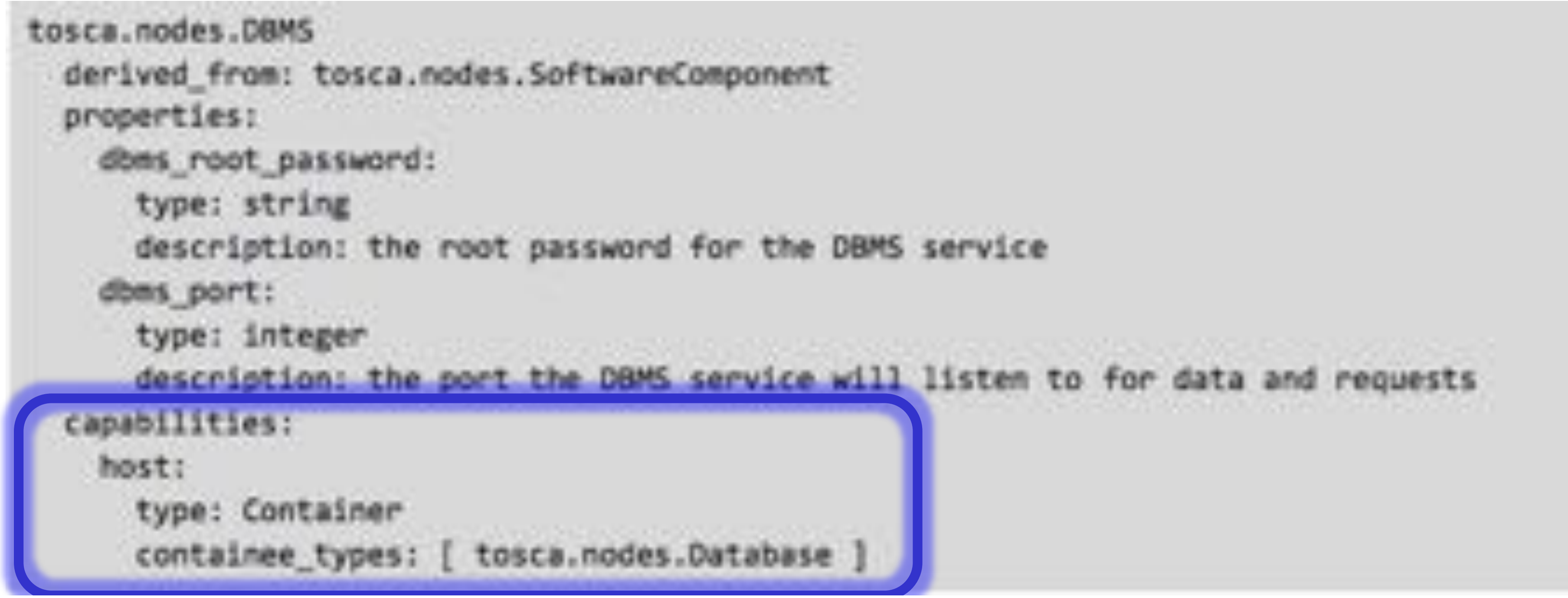
- Describes a Cloud or Software type (e.g. Server or Apache)
- Maps the type to the actual impl. of the lifecycle interface



```
tosca.interfaces.node.Lifecycle:
  create:
    description: Basic lifecycle create operation.
  configure:
    description: Basic lifecycle configure operation.
  start:
    description: Basic lifecycle start operation.
  stop:
    description: Basic lifecycle stop operation.
  delete:
    description: Basic lifecycle delete operation.
```

Node Type (cont.)

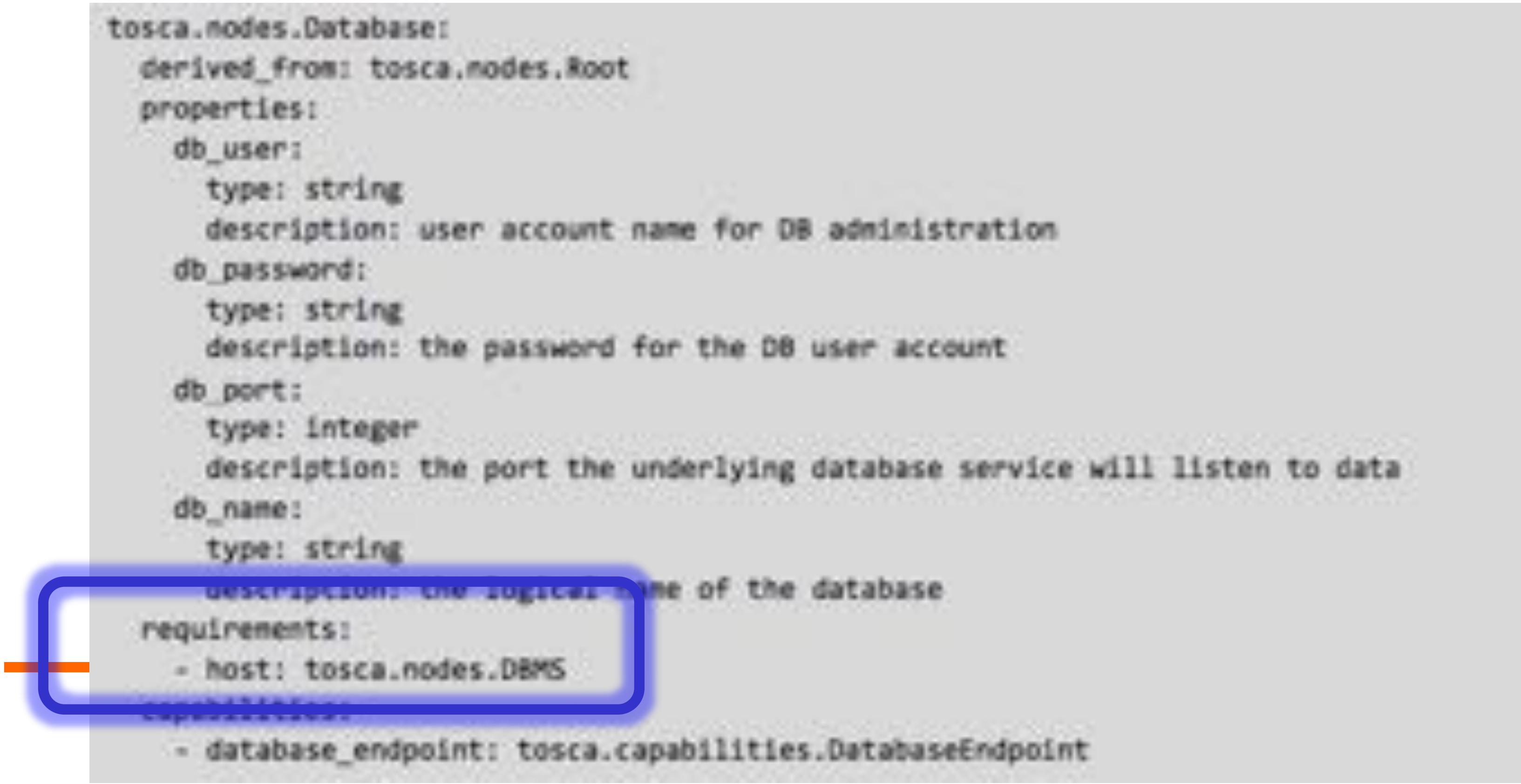
- Defines properties as YAML maps
- Might define capabilities (What it can provide to other nodes)



```
tosca.nodes.DBMS
  derived_from: tosca.nodes.SoftwareComponent
  properties:
    dbms_root_password:
      type: string
      description: the root password for the DBMS service
    dbms_port:
      type: integer
      description: the port the DBMS service will listen to for data and requests
  capabilities:
    host:
      type: Container
      containee_types: [ tosca.nodes.Database ]
```

Node Type (cont.)

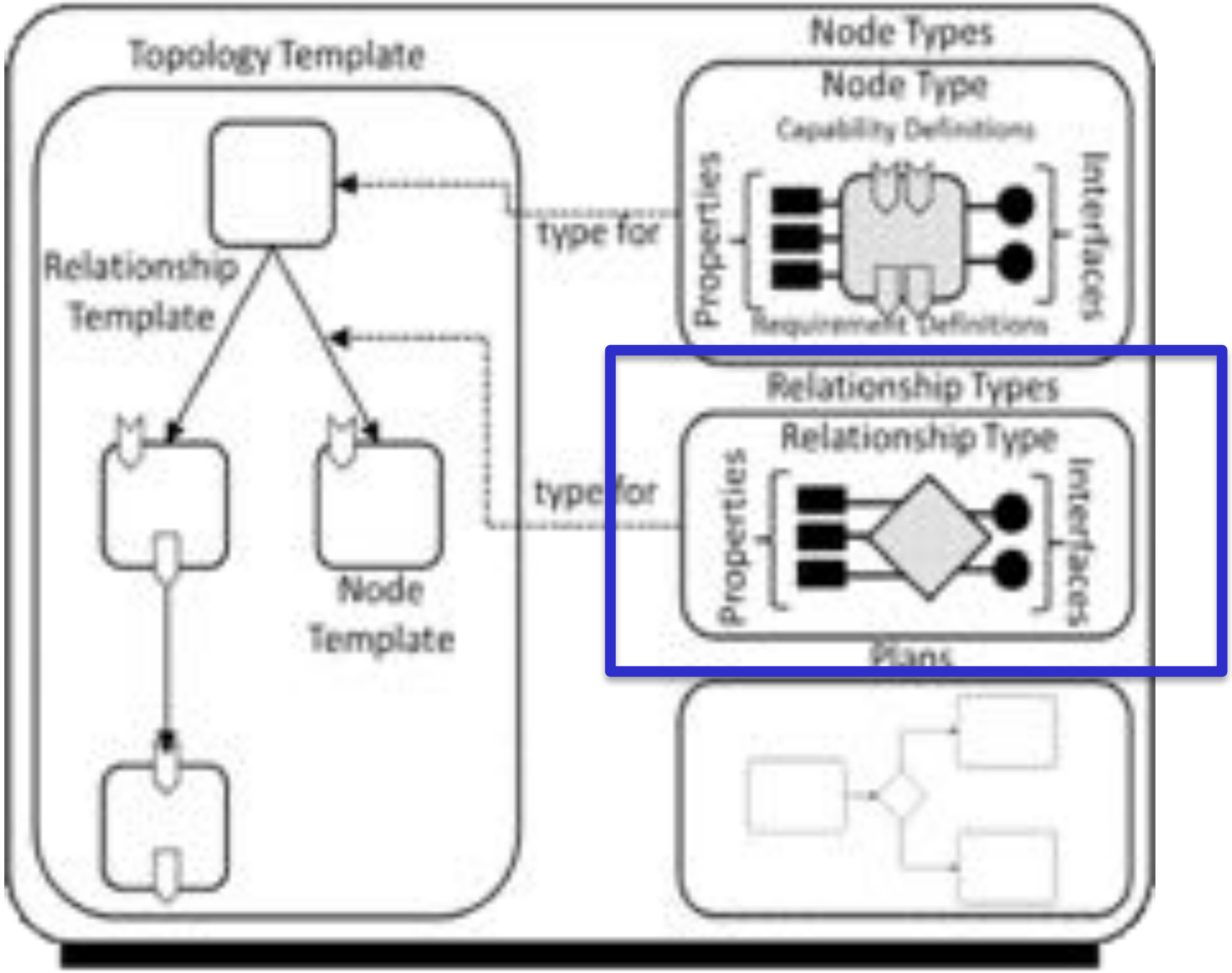
- Might define requirements (what it needs from other nodes)



```
tosca.nodes.Database:
  derived_from: tosca.nodes.Root
  properties:
    db_user:
      type: string
      description: user account name for DB administration
    db_password:
      type: string
      description: the password for the DB user account
    db_port:
      type: integer
      description: the port the underlying database service will listen to data
    db_name:
      type: string
      description: the logical name of the database
  requirements:
    - host: tosca.nodes.DBMS
  capabilities:
    - database_endpoint: tosca.capabilities.DatabaseEndpoint
```

TOSCA Core Ingredients

TOSCA Service Template



Relationship Type

- Requirements and Capabilities are an implicit way to describe relationships
- Usually you need the explicit way
  - You need hooks to configure the source or target node or both
- So relationships have types and interfaces as well

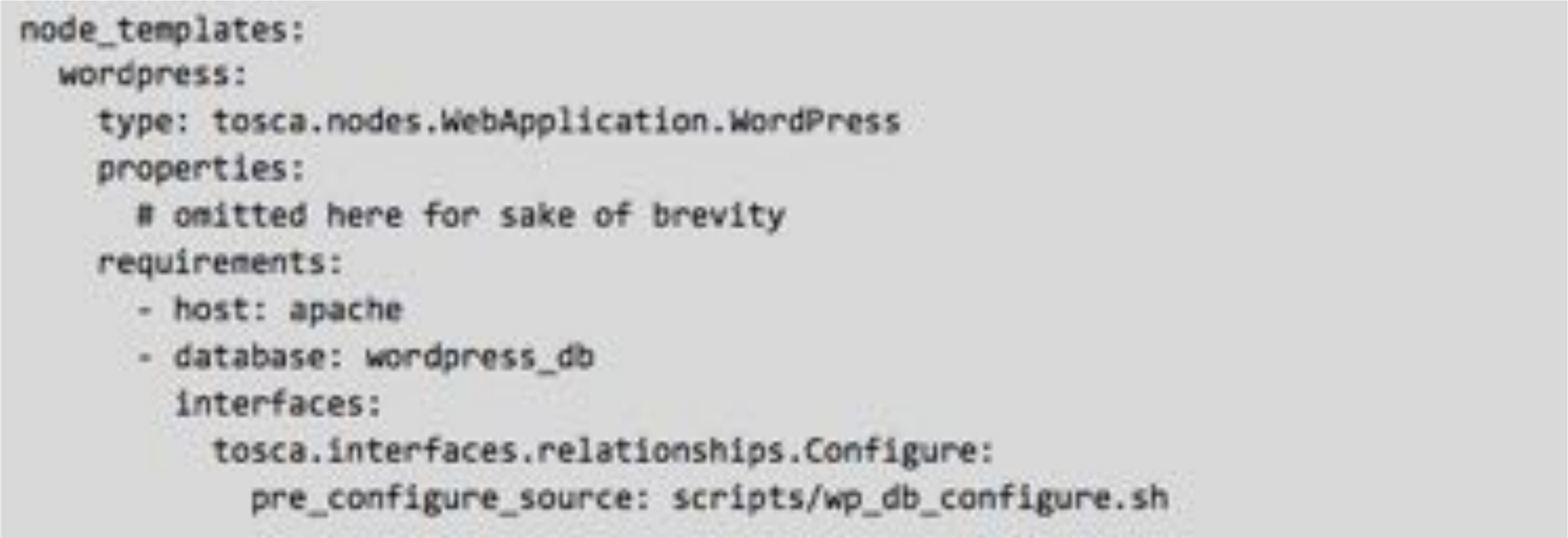
Relationships (cont.)

- The basic relationship types are:
  - **dependsOn** – abstract type and its sub types:
  - **hostedOn** – a node is contained within another
  - **connectsTo** – a node has a connection configured to another
- The basic interface is configure
  - **preconfigure_source**, **preconfigure_target**
  - **postconfigure_source**, **postconfigure_target**
  - **add_target**, **remove_target**

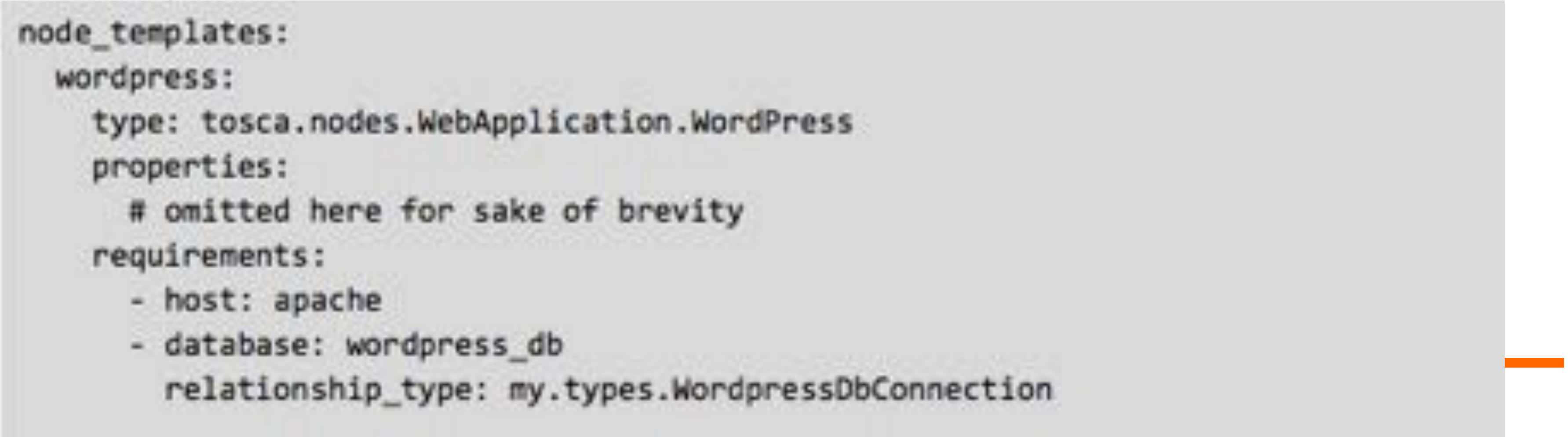
Node Templates

- An instance of a type (like Object to Class)
- Has specific properties
- Has artifacts:
  - What to install
  - How to install (mapped to interface hooks)
- Has requirements and capabilities (or relationships)

Node Template (Examples)

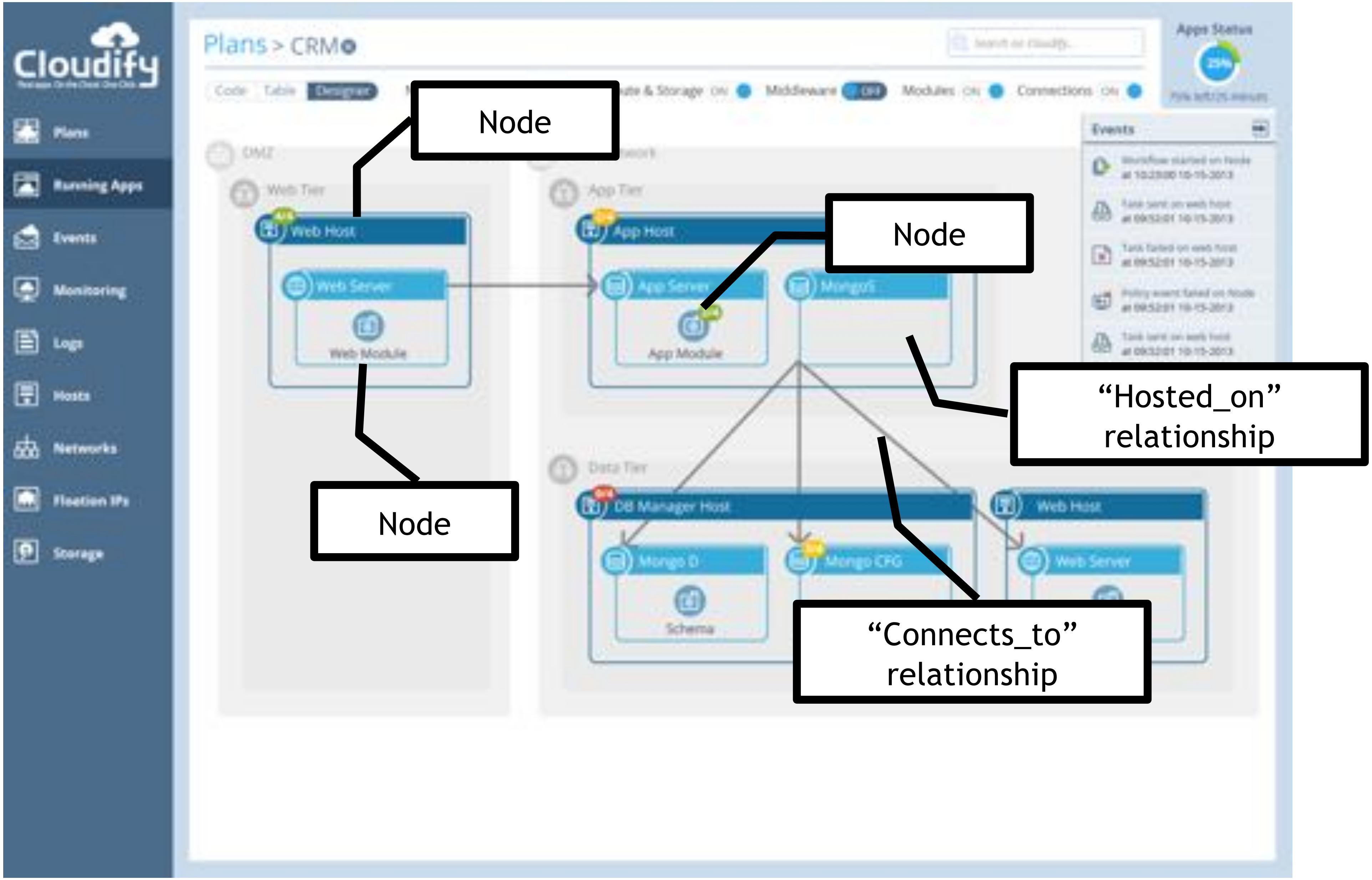


```
node_templates:
  wordpress:
    type: tosca.nodes.WebApplication.WordPress
    properties:
      # omitted here for sake of brevity
    requirements:
      - host: apache
      - database: wordpress_db
        interfaces:
          tosca.interfaces.relationships.Configure:
            pre_configure_source: scripts/wp_db_configure.sh
```



```
node_templates:
  wordpress:
    type: tosca.nodes.WebApplication.WordPress
    properties:
      # omitted here for sake of brevity
    requirements:
      - host: apache
      - database: wordpress_db
        relationship_type: my.types.WordpressDbConnection
```

Translated to TOSCA



Node

Node

Node

“Hosted_on” relationship

“Connects_to” relationship

Workflows

- Imperative flow algorithm
- Using a workflow engine
- Timing the invocation of operations on different node
- **Examples?** Any BPMN specification!

- *But… Considered out of scope for the standard (but currently debated, two factions formed in the TOSCA TC)*

Policies

- Brings monitoring to the orchestration as input
- Ongoing evaluation of Rules
- Enforce SLA, Health, and anything else
- Can invoke more processes
- **Standard Structure:** <Event><Condition><Action>
- **Standard Types:**
  - Access-Control;
  - Placement;
  - QoS (Quality) or (Continuity) CoS;
- **Example?**

TOSCA Policy Example

Event Type

```
<event_type_name>:
  derived_from: <parent_event_type>
  version: <version_number>
  description: <policy_description>
```

Policy Definition

```
<policy_name>:
  type: <policy_type_name>
  description: <policy_description>
  properties:       <property_definitions>
  # allowed targets for policy association
  targets: [ <list_of_valid_target_templates> ]  *
  triggers:
    <trigger_symbolic_name_1>:
        event: <event_type_name>
        # TODO: Allow a TOSCA node filter here
        # required node (resource) to monitor
          target_filter:
          node: <node_template_name> <node_type>
          # Used to reference another node related to
          # the node above via a relationship
          requirement: <requirement_name>
          # optional capability within node to monitor
          capability: <capability_name>
        # required clause that compares an attribute
        # with the identified node or capability
        # for some condition
        condition: <constraint_clause>
        action:
          # a) Define new TOSCA normative strategies
          # per-policy type and use here OR
          # b) allow domain-specific names
          <operation_name>: # (no lifecycle)
            # TBD: Do we care about validation of types?
            # If so, we should use a TOSCA Lifecycle type
            description: <optional description>
            inputs: <list of property assignments >
            implementation: <script> | <service_name>
    <trigger_symbolic_name_2>:
    ...
    <trigger_symbolic_name_n>:
```

Event
name of a normative TOSCA Event Type

Condition
described as a constraint of an attribute of the node (or capability) identified by the filter.

Action
Describes either:
a)a well-known strategy
b)an implementation artifact (e.g., scripts, service) to invoke

*with optional property definitions as **inputs** (to either choice)*

Putting it All Together

```
tosca_definitions_version: tosca_simple_yaml_1_0_0

description: >
  This TOSCA simple profile deployes nodejs, mongodb, elasticsearch, logstash and kibana each on a separate server
  with monitoring enabled for nodejs server where a sample nodejs application is running. The syslog and collectd are
  insatlled on a nodejs server.

imports:
  - tosca_base_type_definition.yaml
  - paypalpizzastore_nodejs_app.yaml
  - elasticsearch.yaml
  - logstash.yaml
  - kibana.yaml
  - collectd.yaml
  - rsyslog.yaml

dsl_definitions:
    host_capabilities: &host_capabilities
      # container properties (flavor)
      disk_size: 10 GB
      num_cpus: { get_input: my_cpus }
      mem_size: 4096 MB
    os_capabilities: &os_capabilities
      architecture: x86_64
      type: Linux
      distribution: Ubuntu
      version: 14.04

topology_template:
  inputs:
    my_cpus:
      type: integer
      description: Number of CPUs for the server.
      constraints:
        - valid_values: [ 1, 2, 4, 8 ]
  ...
```

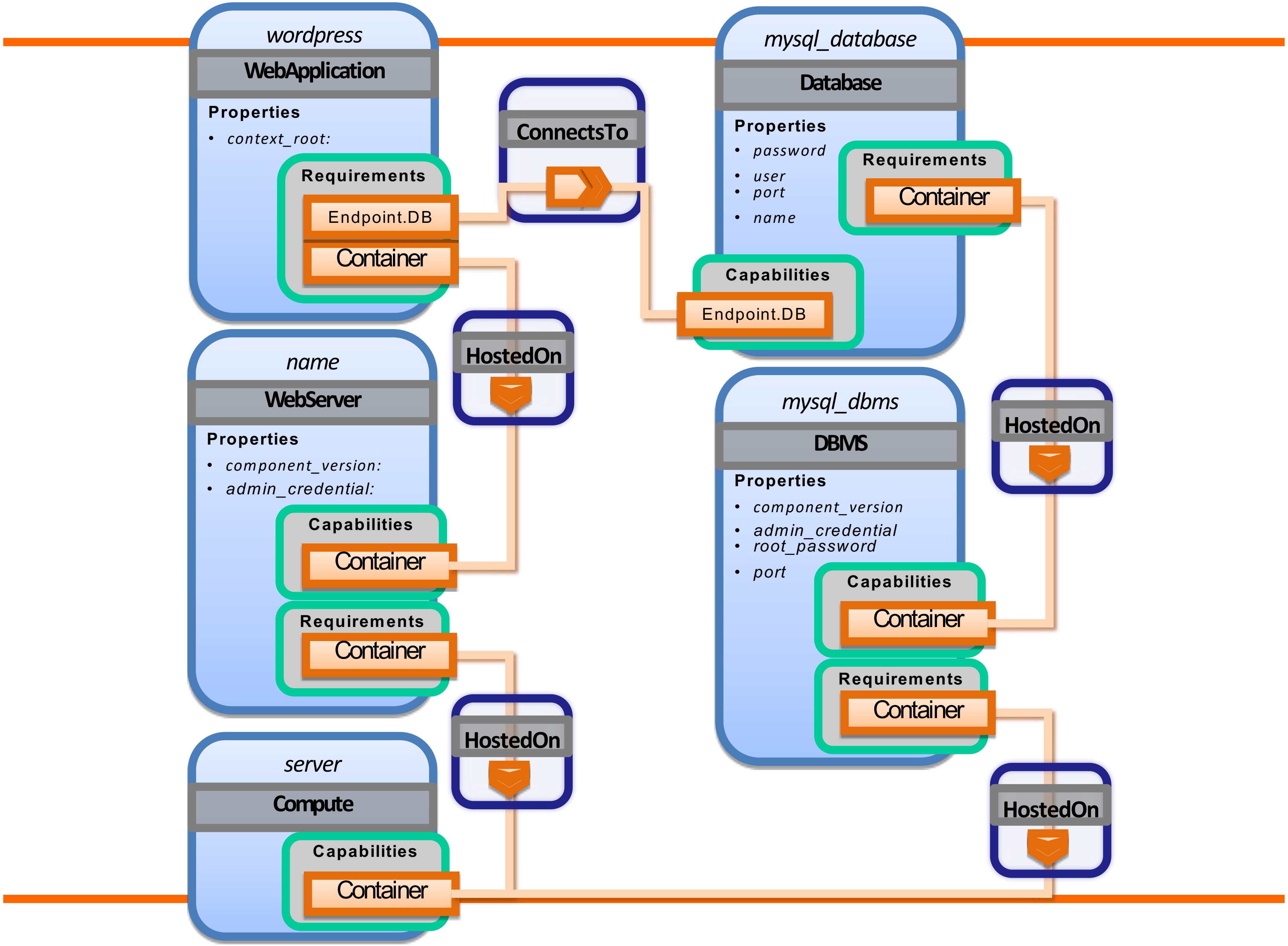
- TOSCA Template contains:
  - Application Topology
    - Nodes
      - Interfaces
      - Properties
      - Artifacts (Plugins in Cloudify)
    - Relationships
      - Interfaces
  - Workflows
  - Policies

TOSCA YAML IasC Examples

- WordPress+MySQL
- NodeJS App+MongoDB

WebServer-DBMS-1:
WordPress - MySQL

JADS Jheronimus Academy of Data Science



wordpress

WebApplication

Properties
- *context_root:*

Requirements
- Endpoint.DB
- Container

ConnectsTo

mysql_database

Database

Properties
- *password*
- *user*
- *port*
- *name*

Requirements
- Container

Capabilities
- Endpoint.DB

HostedOn

name

WebServer

Properties
- *component_version:*
- *admin_credential:*

Capabilities
- Container

Requirements
- Container

HostedOn

mysql_dbms

DBMS

Properties
- *component_version*
- *admin_credential*
- *root_password*
- *port*

Capabilities
- Container

Requirements
- Container

HostedOn

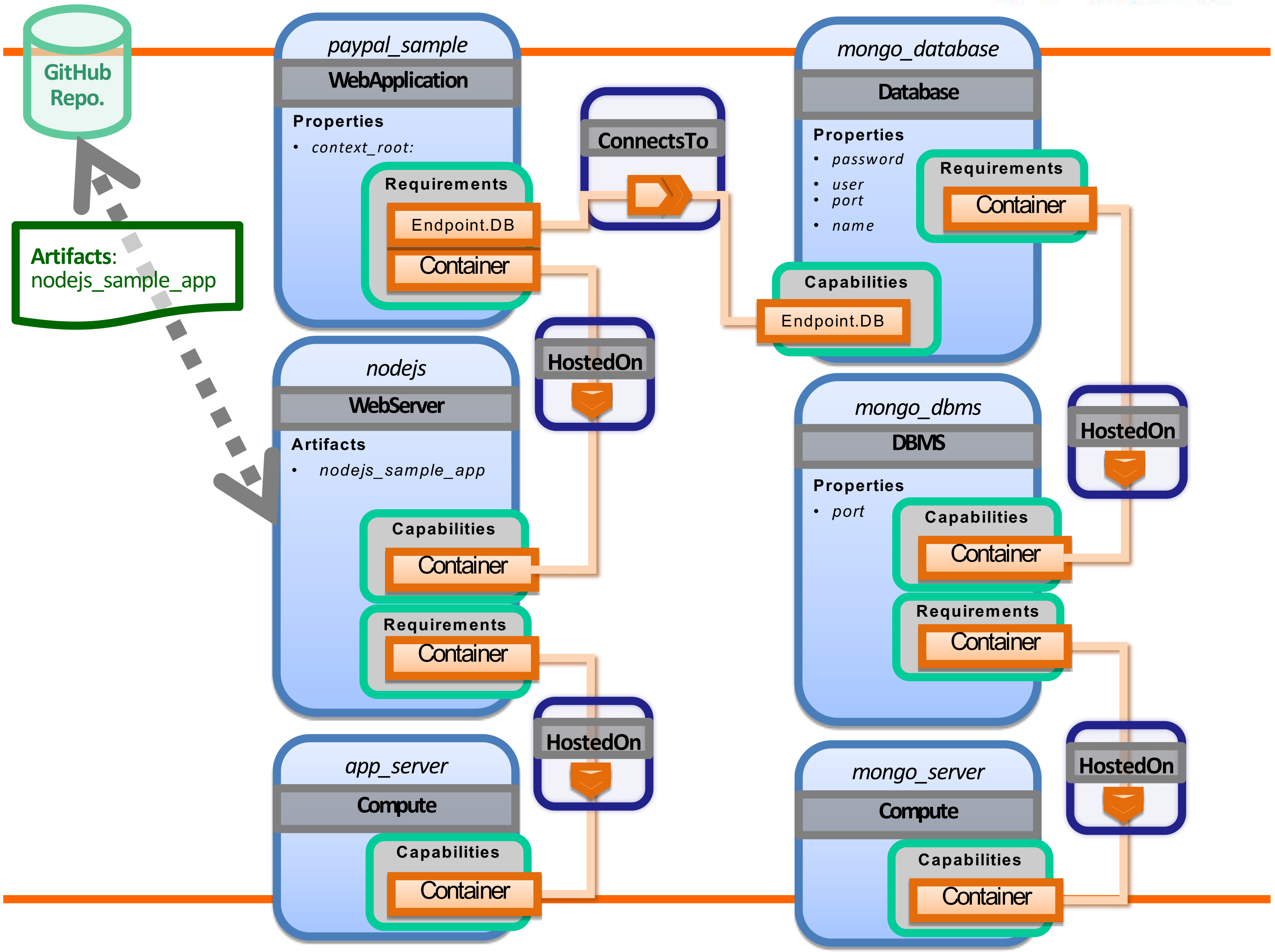
HostedOn

server

Compute

Capabilities
- Container

WebServer-DBMS-3:
Nodejs - MongoDB



Continuous Architecting… Where does TOSCA fit in?

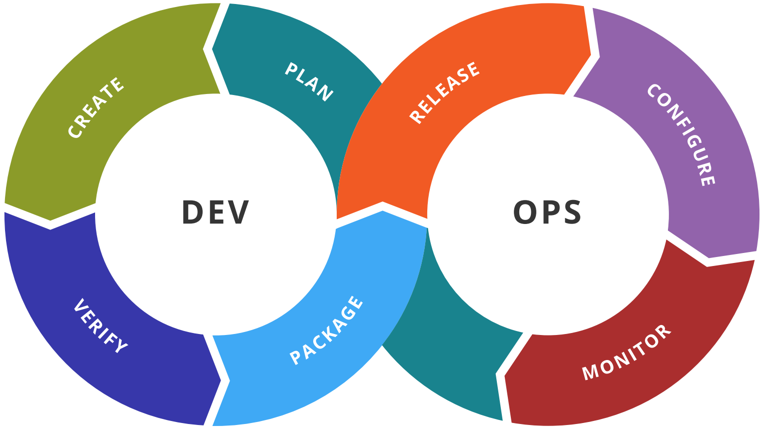
- **The Simple Way** *(more or less ☺)*
  - Capture automated decision-making policies as TOSCA policies, and *let the Orchestrator make your Architecture Decisions*
  - Continuously Evaluate Decisions against SLAs (e.g., Monitoring + Runtime *Instance-Model* Checking)
  - Instrument DevOps Pipeline to *measure* the quantities and qualities of Automated Decision-Making Policies in the blueprint ➔ Continuous Improvement!

Continuous Architecting… Where does TOSCA fit in?

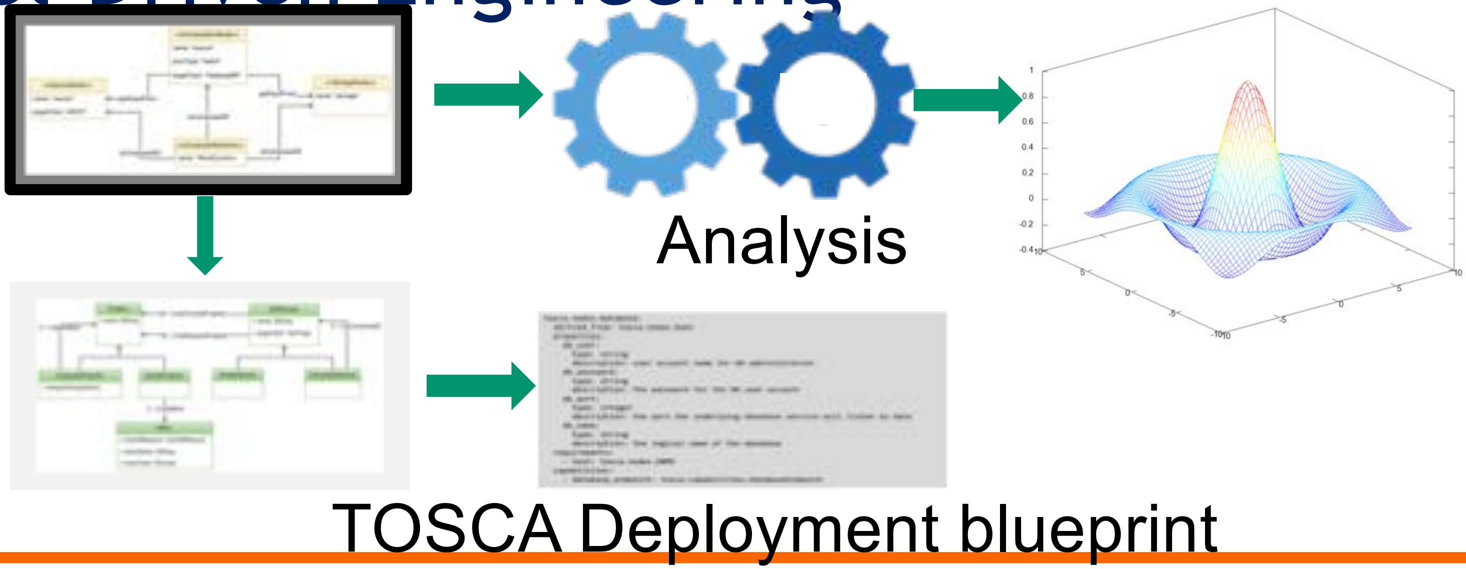
- **The Hard Way…**
  - Use TOSCA blueprint to design the (1) organizational structure of the DevOps team, the (2) Software Architecture it maintains *and* the (3) DevOps pipeline;
  - Use TOSCA-based orchestration to:
    - Study the performance of (1) - (3) for continuous improvement;
    - Use TOSCA-based orchestration automation to make improvement as automated as possible;

Continuous Architecting with TOSCA: the EU H2020 DICE Example

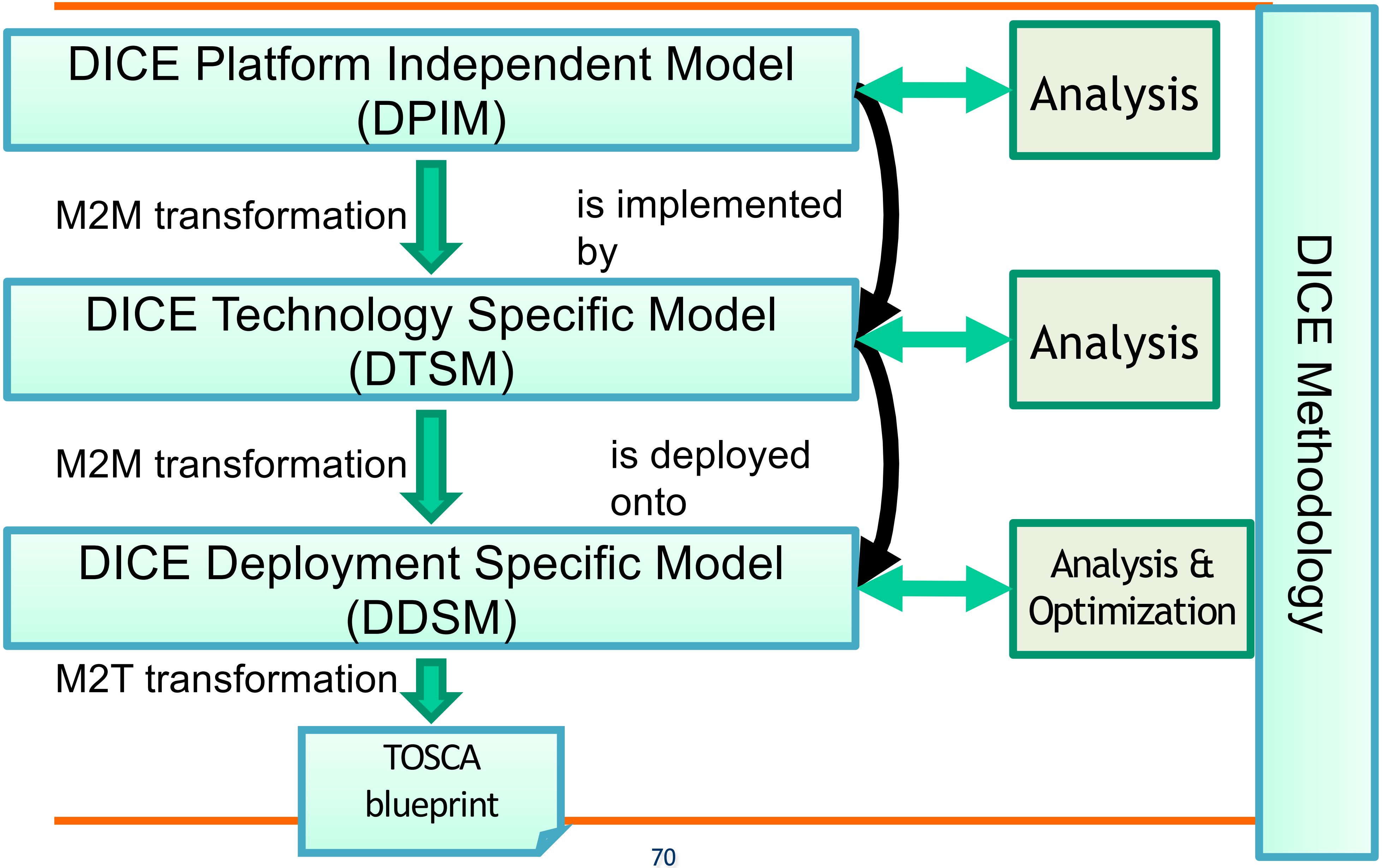
- DevOps



- Model-Driven Engineering



DICER, incremental arch. modeling and analysis towards TOSCA



DICE Platform Independent Model (DPIM)

M2M transformation

is implemented by

DICE Technology Specific Model (DTSM)

M2M transformation

is deployed onto

DICE Deployment Specific Model (DDSM)

M2T transformation

TOSCA blueprint

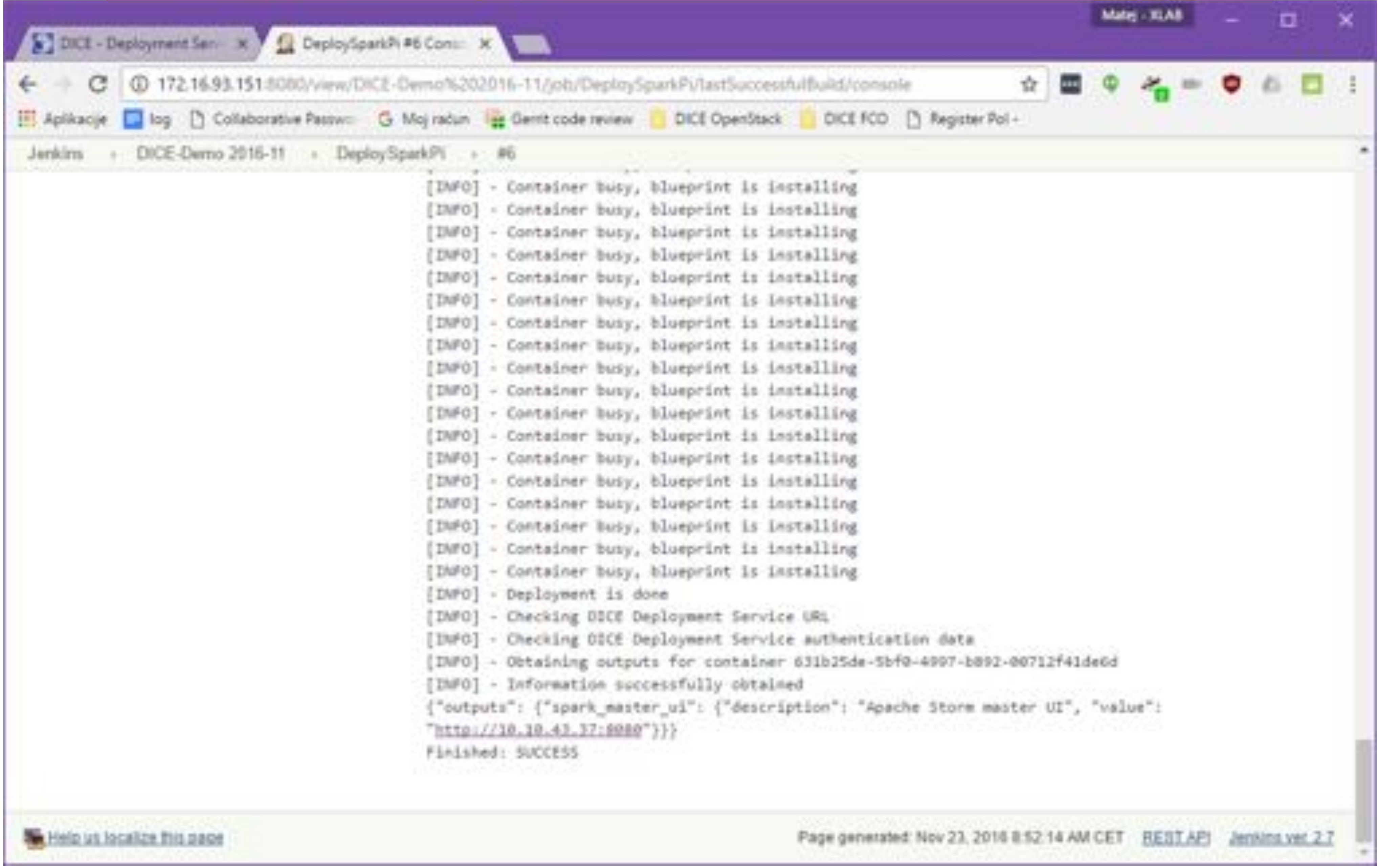
Analysis

Analysis

Analysis & Optimization

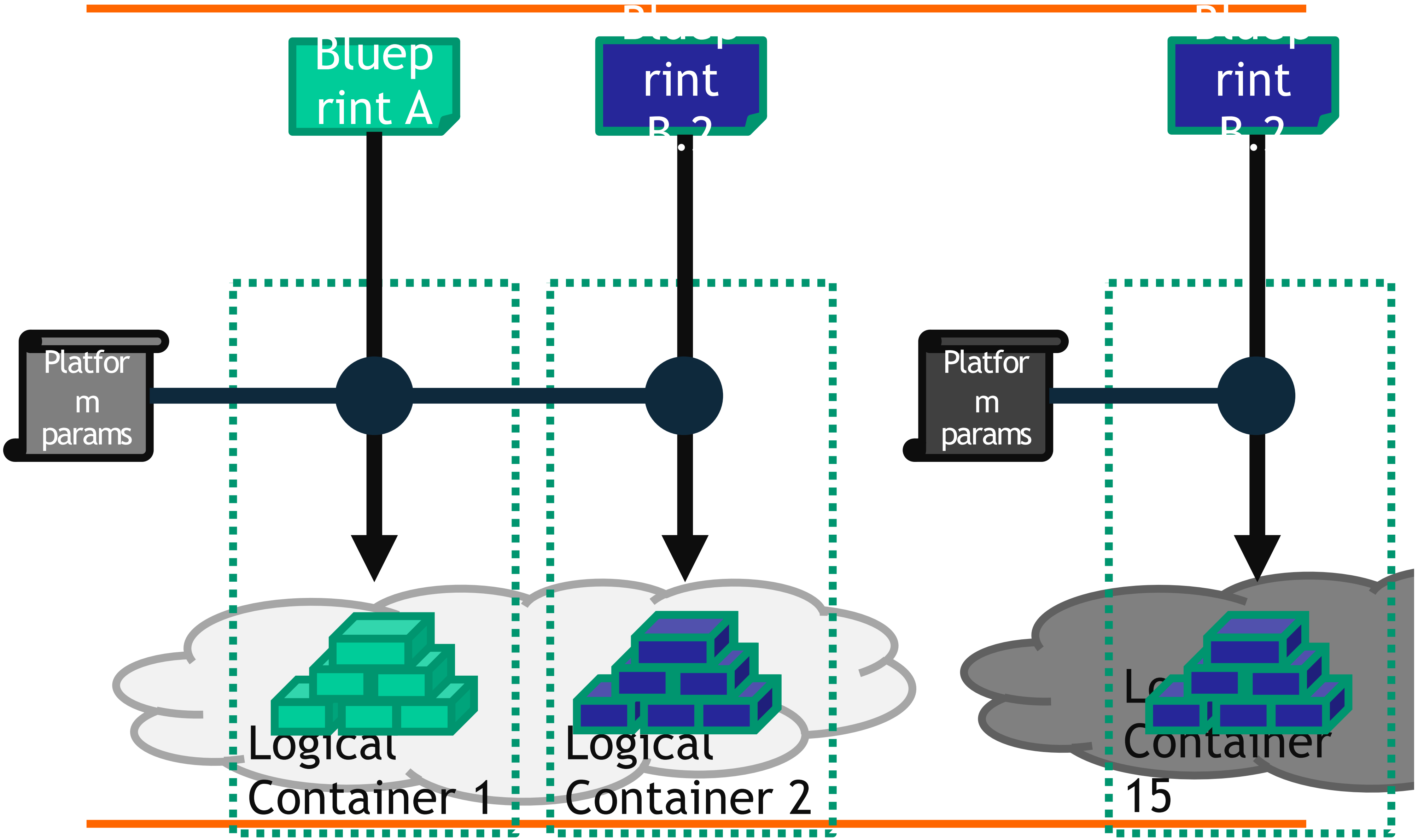
DICE Methodology

DICER actual deployment



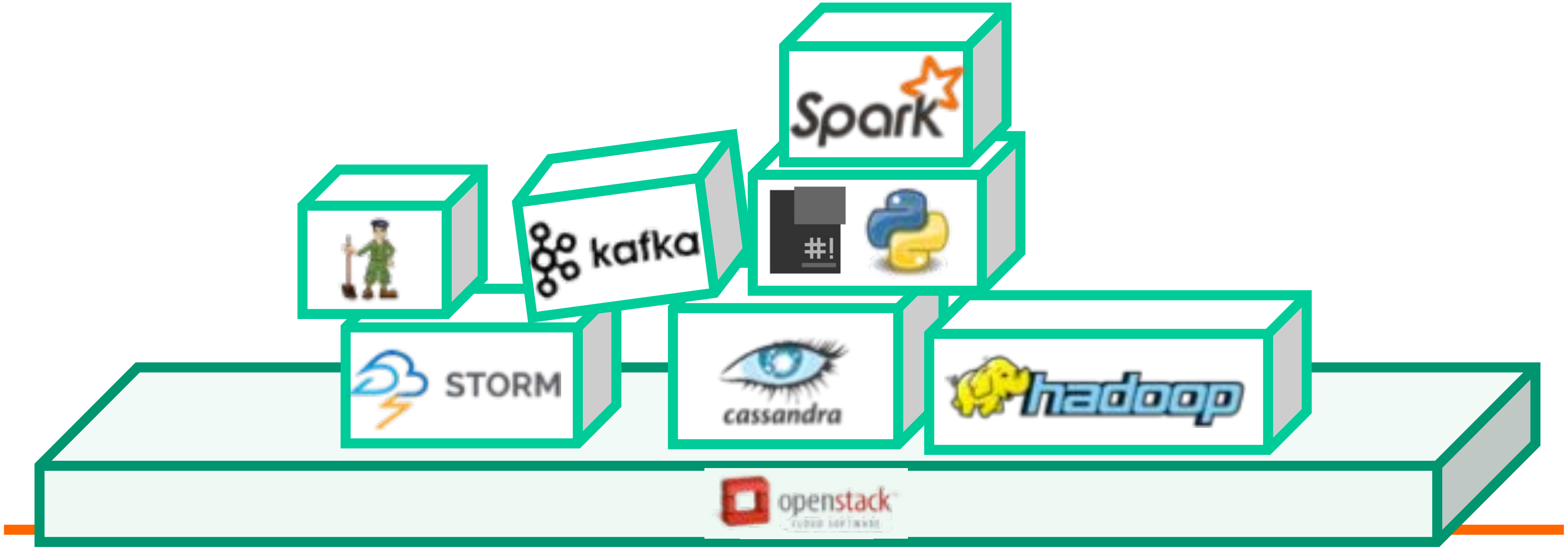
```
[INFO] - Container busy, blueprint is installing
[INFO] - Container busy, blueprint is installing
[INFO] - Container busy, blueprint is installing
[INFO] - Container busy, blueprint is installing
[INFO] - Container busy, blueprint is installing
[INFO] - Container busy, blueprint is installing
[INFO] - Container busy, blueprint is installing
[INFO] - Container busy, blueprint is installing
[INFO] - Container busy, blueprint is installing
[INFO] - Container busy, blueprint is installing
[INFO] - Container busy, blueprint is installing
[INFO] - Container busy, blueprint is installing
[INFO] - Container busy, blueprint is installing
[INFO] - Container busy, blueprint is installing
[INFO] - Container busy, blueprint is installing
[INFO] - Container busy, blueprint is installing
[INFO] - Container busy, blueprint is installing
[INFO] - Container busy, blueprint is installing
[INFO] - Deployment is done
[INFO] - Checking DICE Deployment Service URL
[INFO] - Checking DICE Deployment Service authentication data
[INFO] - Obtaining outputs for container 631b25de-5bf0-4997-b892-00712f41de6d
[INFO] - Information successfully obtained
{"outputs": {"spark_master_ui": {"description": "Apache Storm master UI", "value":
"http://10.10.43.37:8080"}}}
Finished: SUCCESS
```

DICER Delivery Service

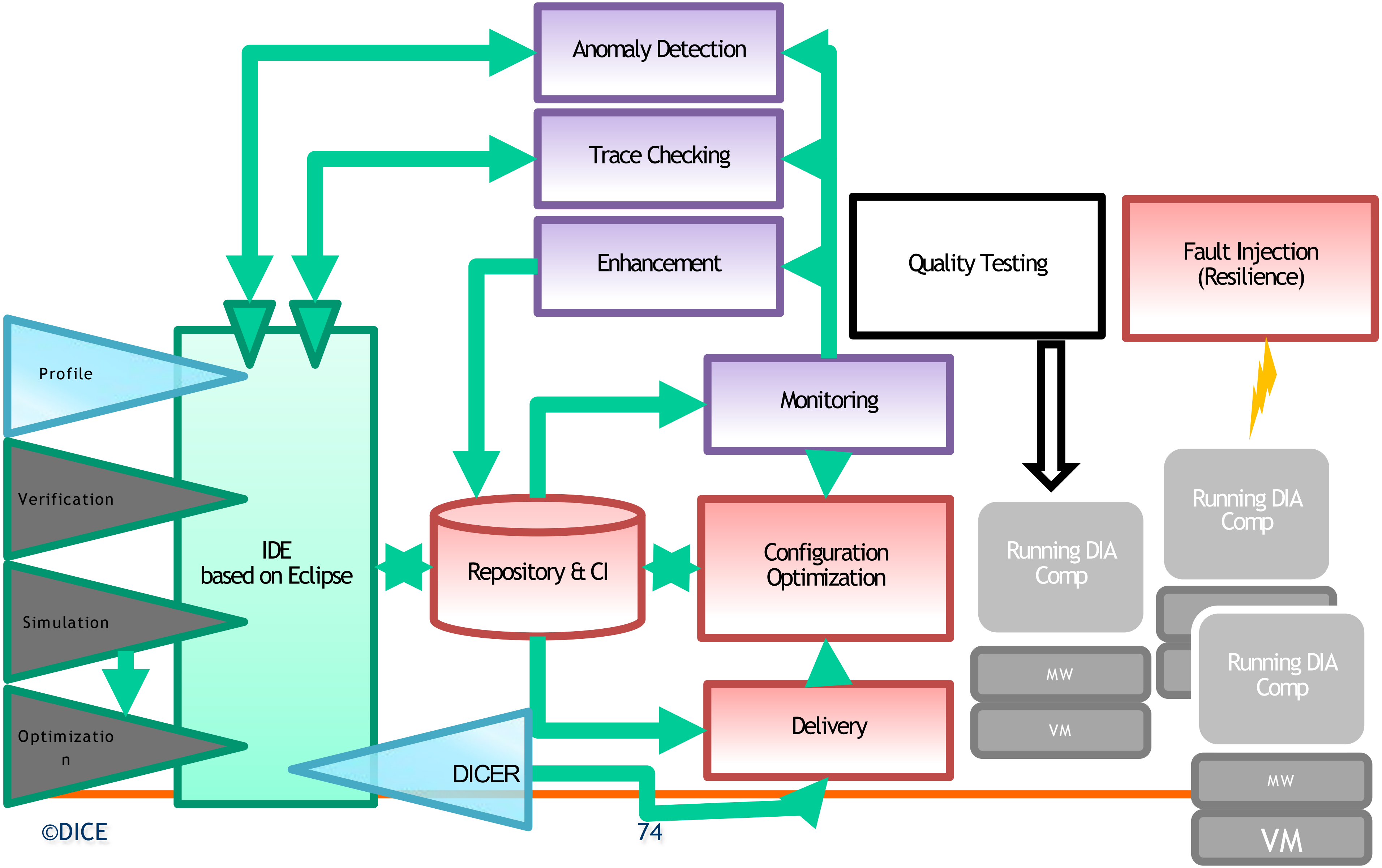


DICER TOSCA technology library

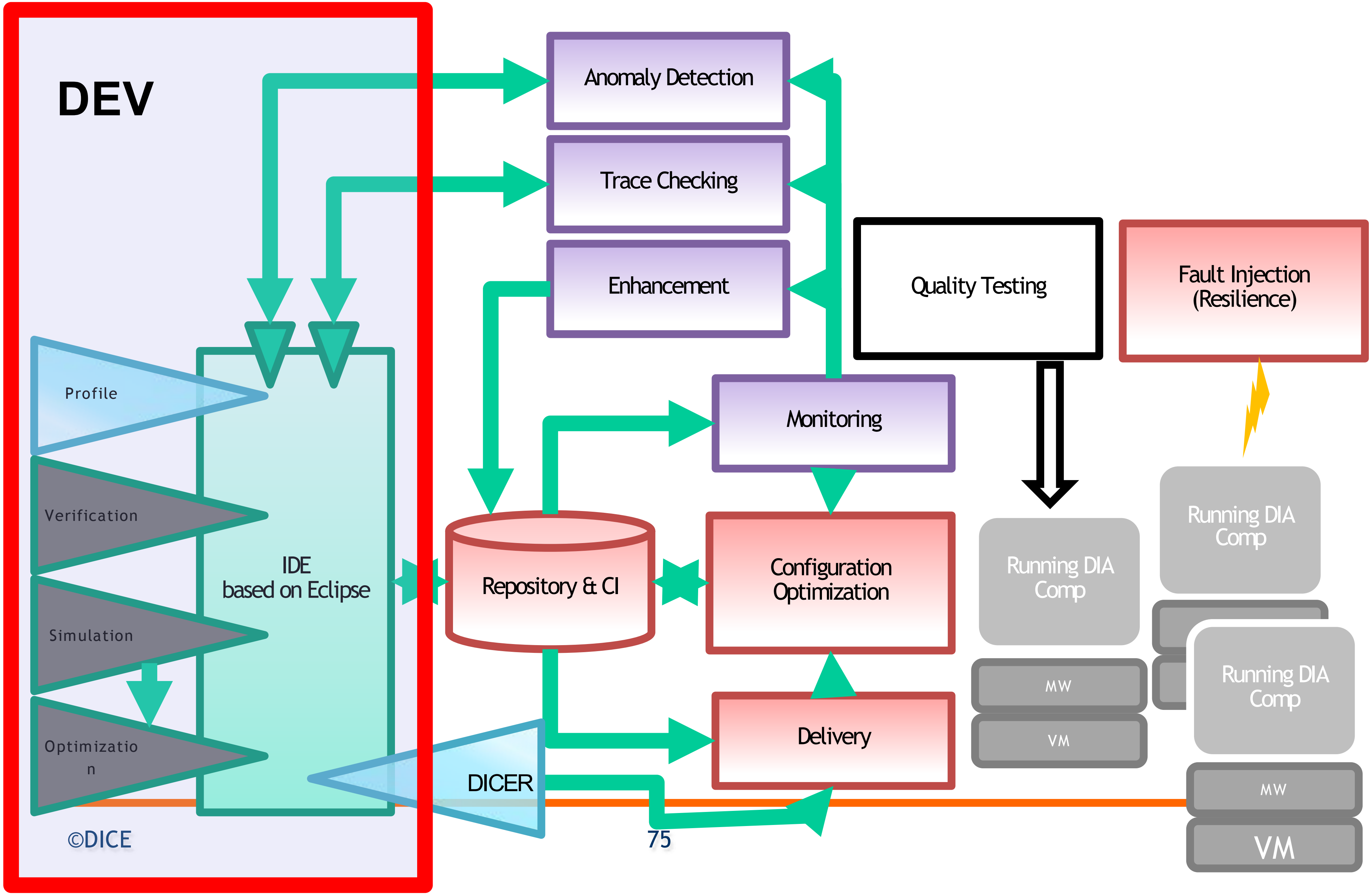
- A plug-in for Cloudify
- A single import line in the TOSCA blueprint
- Node types + Chef cookbooks for Major Big Data services
- Unified across supported IaaS vendors



And After Deployment?

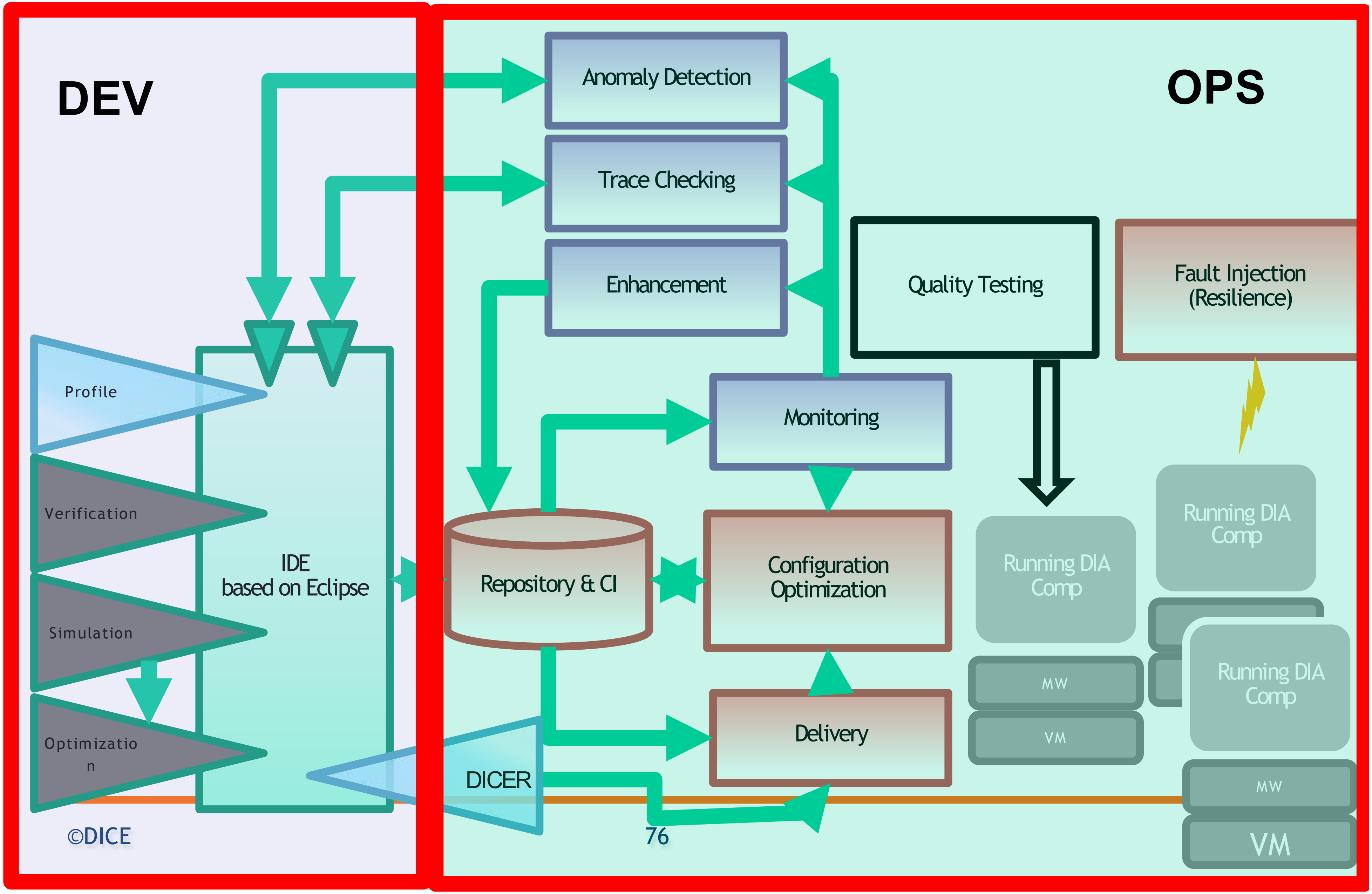


And After Deployment?



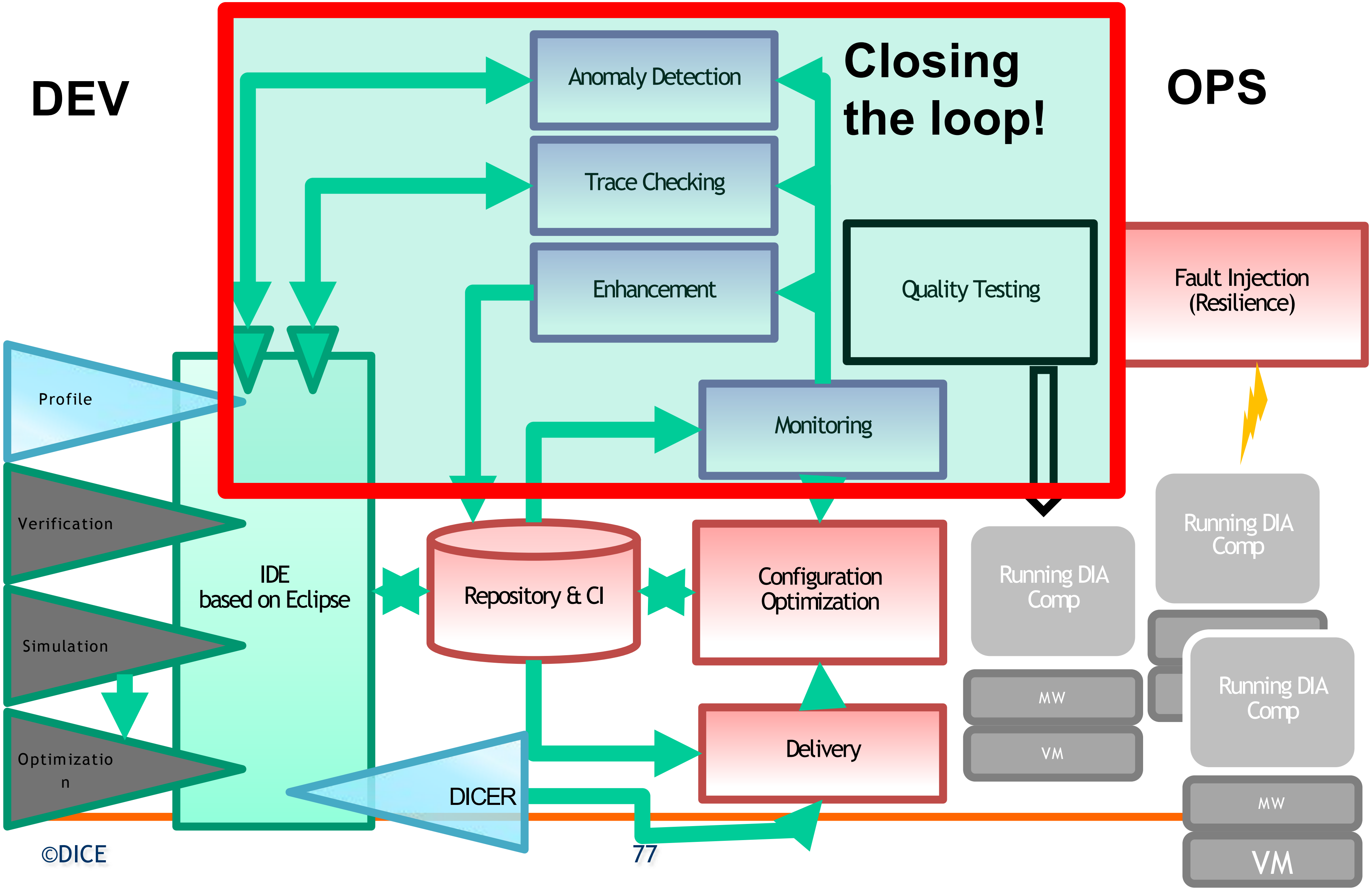
©DICE

And After Deployment?



©DICE

And After Deployment?



©DICE

Conclusions (1)

- Continuous Architecting ➔ New arch. drivers:
  - Modifiability
  - Observability
  - Organisability
  - Speed
  - Failure
  - …

Conclusions (1)

- Continuous Architecting ➔ New arch. drivers:
  - **Modifiability**
  - Observability
  - **Organisability**
  - Speed
  - Failure
  - …

Conclusions (1)

- Of these “New” architecture drivers:
  - Modifiability
  - Organisability
  - …
- Only these are Actually New!
  - **Observability**
  - **Speed**
  - **Failure**
  - …

Conclusions (2)

- New architecture drivers:
  - Modifiability
  - Observability
  - Organisability
  - Speed
  - Failure
  - …

- Continuous Architecting – "more of the same, ***only faster***" ➔ TOSCA-centric Software Architecting!

Future Work (2)

- What we miss, architecturally:
  - A better connection between the design of software, the design of the infrastructure, and the design of the organization;
  - A precise and rigorous comparison approach between the new languages and tools for coding infrastructures;
  - Metrics to track and evaluate all of the above;

Conclusions & Future Work (3)

- Who is the Architect?
  - Anyone who enables Continuous-*
  - Anyone who enables an Agile Organisation
  - Anyone who enables for new Arch. Drivers’ equivalent metrics (e.g., Observability, Modifiability)
  - So… The architect is a Community Shepherd
    - → it can be anyone!

“The Architect is my Shepherd […]”

Links

- DICE deployment service: https://github.com/dice-project/DICE-Deployment-Service
- Big Data blueprint examples: https://github.com/dice-project/DICE-Deployment-Examples
- DICER:

  https://github.com/dice-project/DICER

That’s all folks!

Any Questions?

References

[1] Erder, Murat and Pureur, Pierre. Continuous Architecture: Sustainable Architecture in an Agile and Cloud-Centric World. Amsterdam: Morgan Kaufmann, 2016.

[2] Continuous Testing Paperback – January 2, 2014 by W. Ariola, C. Dunlop

[3] Part of the Pipeline: Why Continuous Testing Is Essential, by Adam Auerbach, TechWell Insights August 2015

[4] M. Fowler Continuous Integration, https://www.thoughtworks.com/continuous-integration

[5] *Chen, Lianping (2015)* *"Continuous Delivery: Huge Benefits, but Challenges Too"* *IEEE Software*. ***32*** *(2): 50.*

[6] http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/csd03/TOSCA-Simple-Profile-YAML-v1.0-csd03.html

[7] P. Lipton, D. Palma, M. Rutkowski, and D. A. Tamburri, "Tosca solves big problems in the cloud and beyond!" *IEEE Cloud*, vol. 21, no. 11, pp. 31–39, 2016.

[8] Bengtsson, PerOlof, Lassing, Nico, Bosch, Jan and van Vliet, Hans. "Architecture-level modifiability analysis (ALMA)." *Journal of Systems and Software* 69 , no. 1--2 (2004): 129--147.

[9] Tamburri, D. A.; Lago, P. & van Vliet, H. (2013), 'Uncovering Latent Social Communities in Software Development.', *IEEE Software* **30** (1) , 29-36 .

[10] M. Di Penta, D. A. Tamburri, Combining Quantitative and Qualitative Methods in Empirical Software Engineering Proceedings of the 10th Joint Meeting of the European Software Engineering Conference and the ACM Sigsoft Symposium of the Foundations of Software

Other Biblio

[11] Bass, L. J.; Weber, I. M. & Zhu, L. (2015), *DevOps - A Software Architect's Perspective*. , Addison-Wesley .

[12] Tamburri, D. A. & Nitto, E. D. (2015), When Software Architecture Leads to Social Debt., *in* Len Bass; Patricia Lago & Philippe Kruchten, ed., 'WICSA' , IEEE Computer Society, pp. 61-64 .

[13] Tamburri, D. A.; Kruchten, P.; Lago, P. & van Vliet, H. (2015), 'Social debt in software engineering: insights from industry.', *J. Internet Services and Applications* **6** (1) , 10:1-10:17 .

[14] Tamburri, D. A.; Lago, P. & van Vliet, H. (2013), 'Organizational social structures for software engineering.', *ACM Comput. Surv.* **46** (1) , 3 .