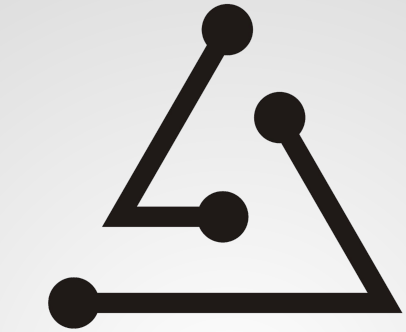
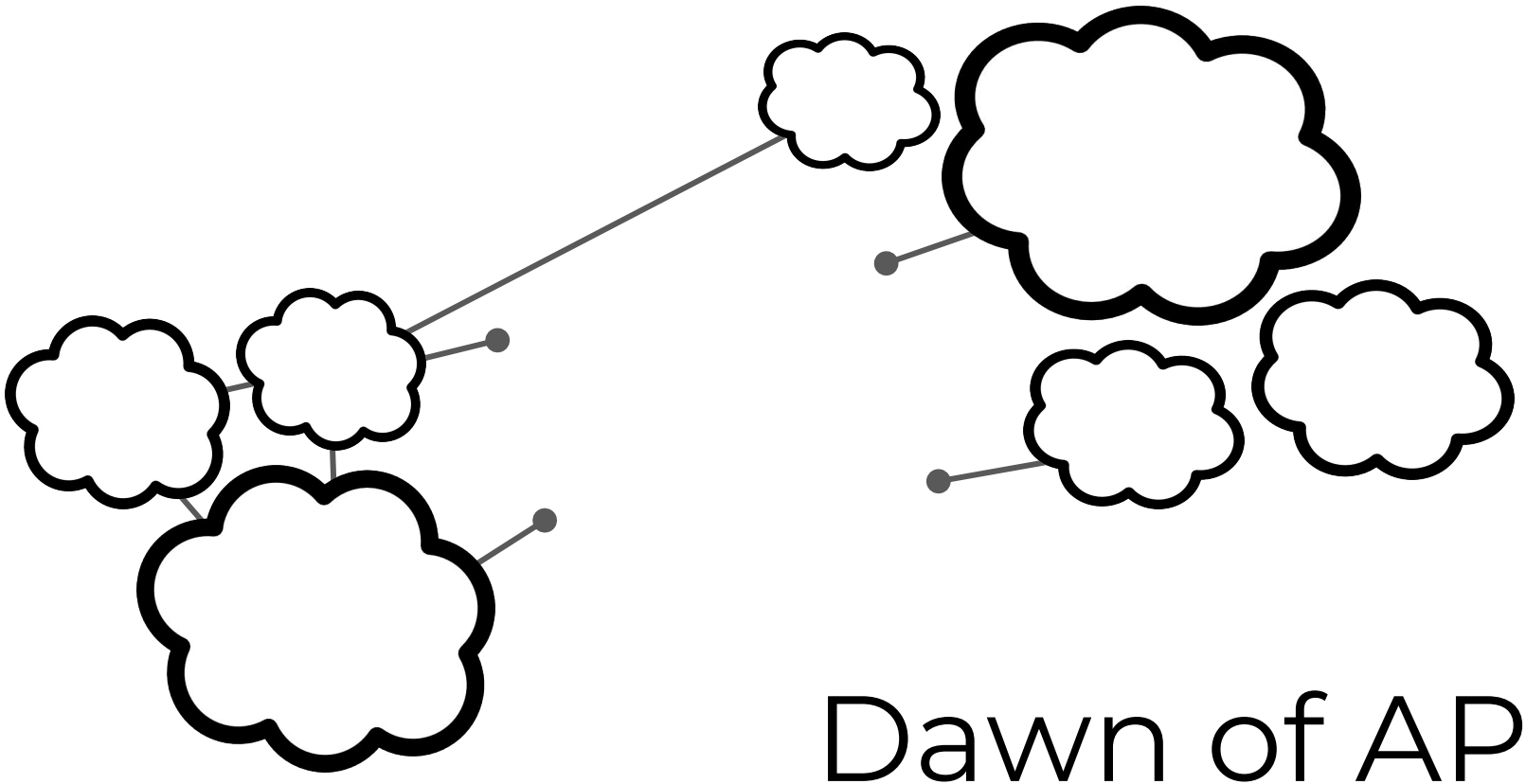


ARIA

Model Driven Orchestration with TOSCA and ARIA

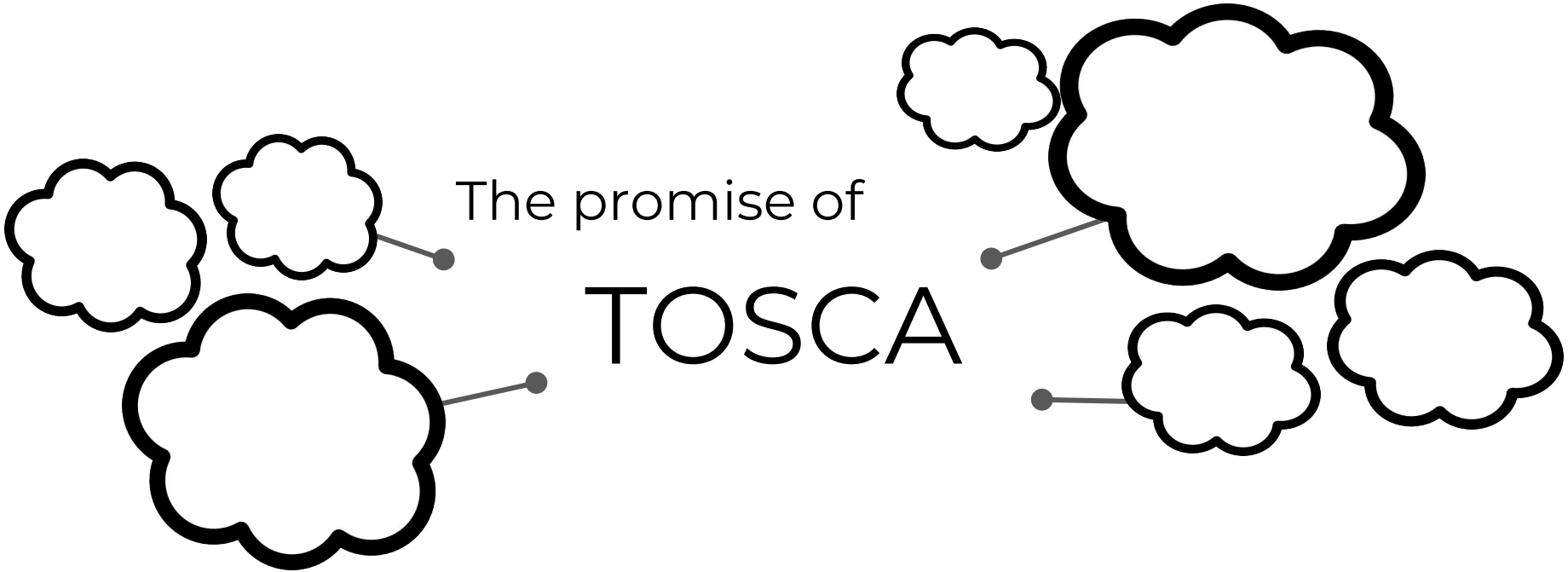


Dawn of APIs

Help

All changes saved in Drive

Arial



Topology and Orchestration Specification for Cloud Applications

TOSCA Descriptive Language

Normative Types

“tosca.nodes.compute”

Abstraction

openstack.nodes.compute:
Derived_from: tosca.nodes.compute

Extensibility

“aria.kubernetes.Microservice”

Composition

Imports: app-backend-blueprint.yaml

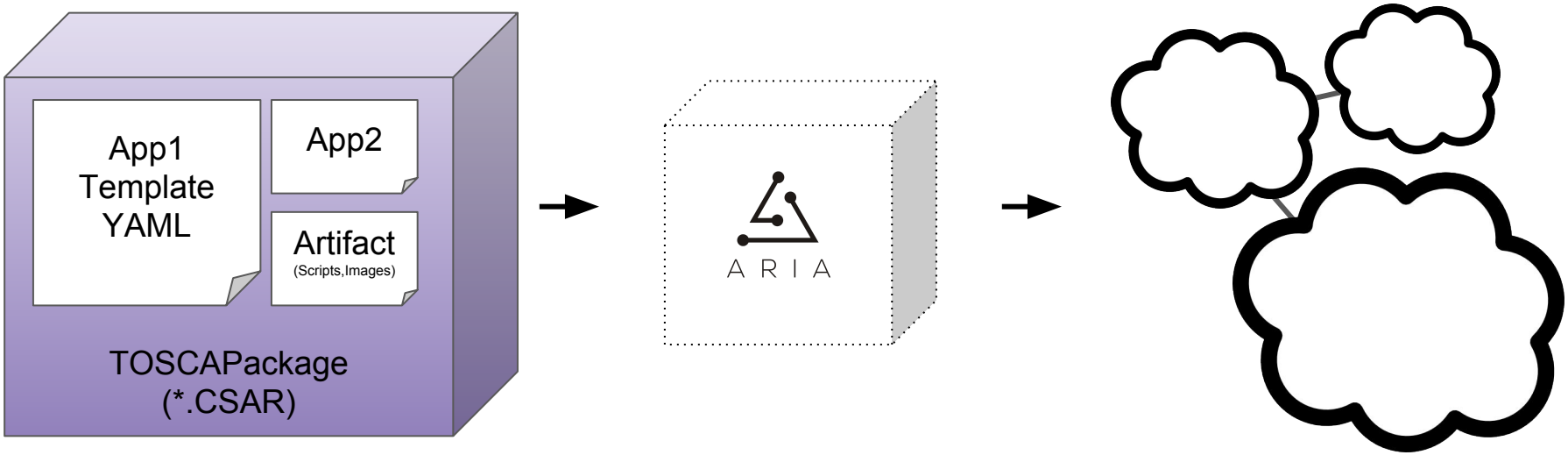
ARIA

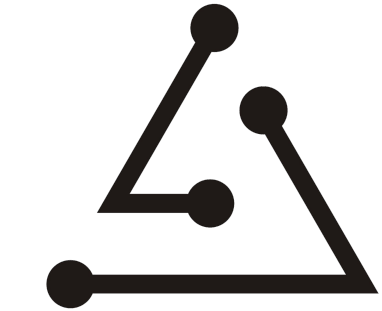
TOSCA Descriptive Language

- Strict
  - Object-oriented, strictly typed, polymorphic
  - Rich set of base types (Simple Profile + Simple Profile for NFV)
  - Normative lifecycle (install, uninstall, start, stop)
- Agnostic
  - Not specific to any cloud provider (multi-VIM is hard)
  - Not specific to any machine technology (tosca.nodes.Compute = VM, container, cage, or...?)
  - Base types designed to be lowest-common denominators (politics)
  - Base types are optional
  - Support for generic workflows in TOSCA 1.1

ARIA

Orchestration of APIs



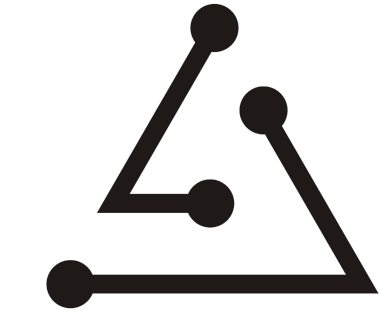


ARIA

Implementation of TOSCA

Open Source

Apache 2.0 License



ARIA

Open Governance

Apache Software Foundation

ARIA's Goal

1. **Simplify** BY 10X **path to support TOSCA**

ARIA

ARIA's Goal

1. Simplify BY path to support TOSCA
2. **Simplify** 10X **creation of TOSCA applications**

ARIA

ARIA's Goal

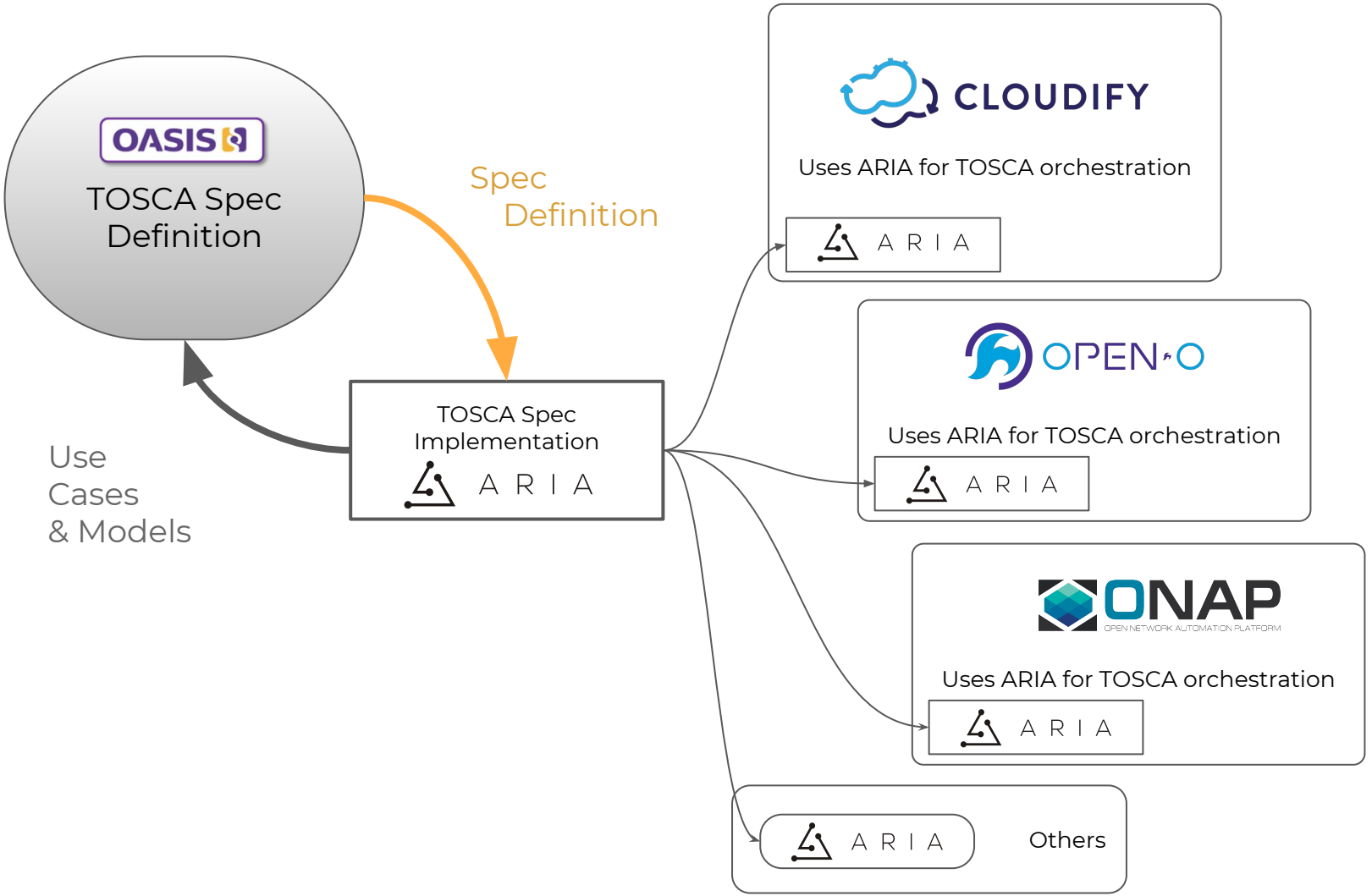
1. Simplify BY path to support TOSCA
2. Simplify creation of TOSCA applications
3. **Simplify** 10X **exploration of new models**

ARIA

Project Principles

- Vendor Neutral
- Technology Independent
- Interoperability across orchestration vendors

ARIA


OASIS
TOSCA Spec Definition
Spec Definition
Use Cases & Models
TOSCA Spec Implementation
ARIA
CLOUDIFY
Uses ARIA for TOSCA orchestration
ARIA
OPEN-O
Uses ARIA for TOSCA orchestration
ARIA
ONAP
OPEN NETWORK AUTOMATION PLATFORM
Uses ARIA for TOSCA orchestration
ARIA
ARIA
Others

So, What is ARIA?

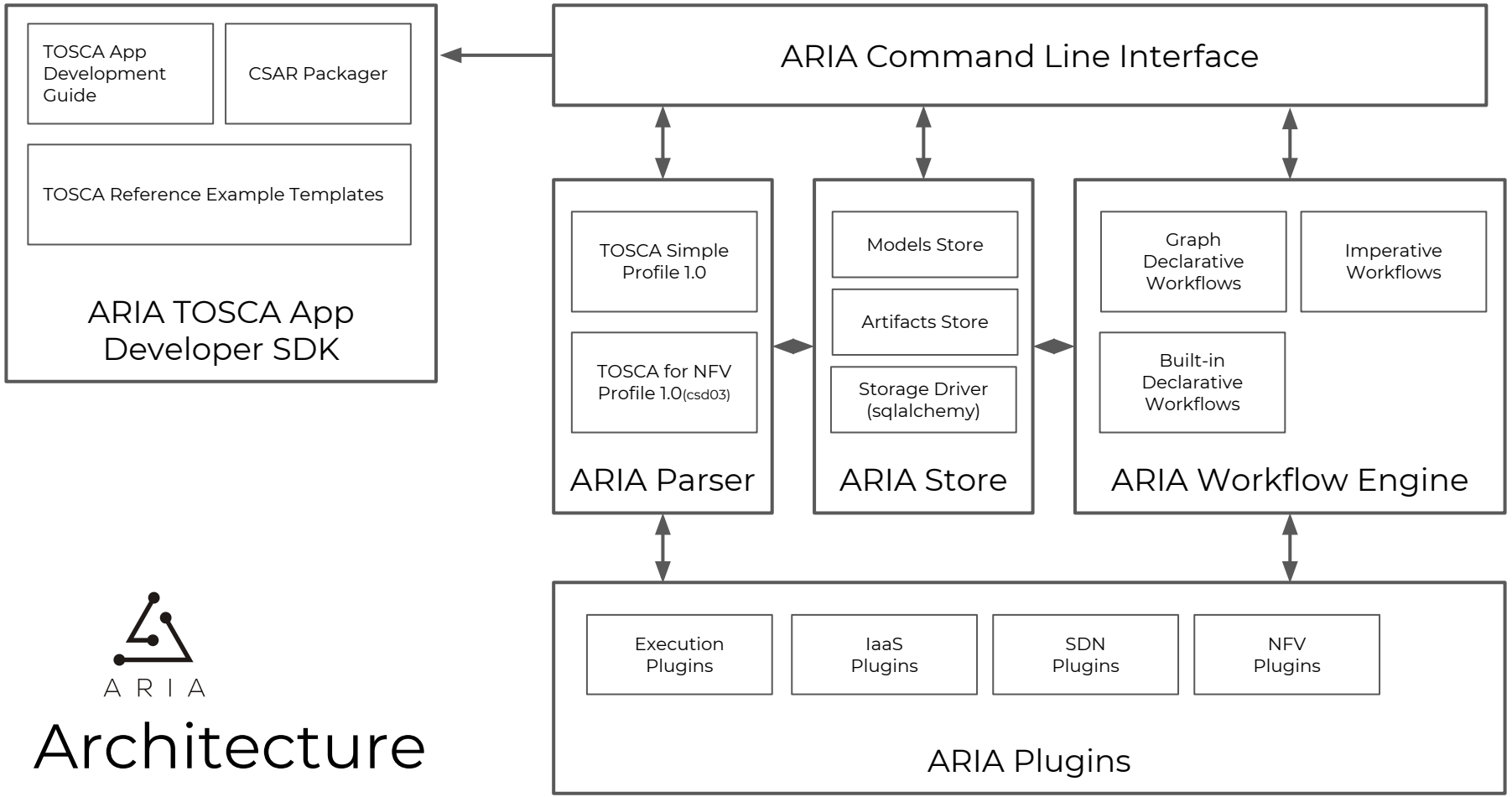
What is ARIA?

1. Python LIBRARY for orchestration products to support TOSCA Profiles

2. TOSCA SDK for creating TOSCA applications

3. CLI TOOL to orchestrate TOSCA templates

ARIA



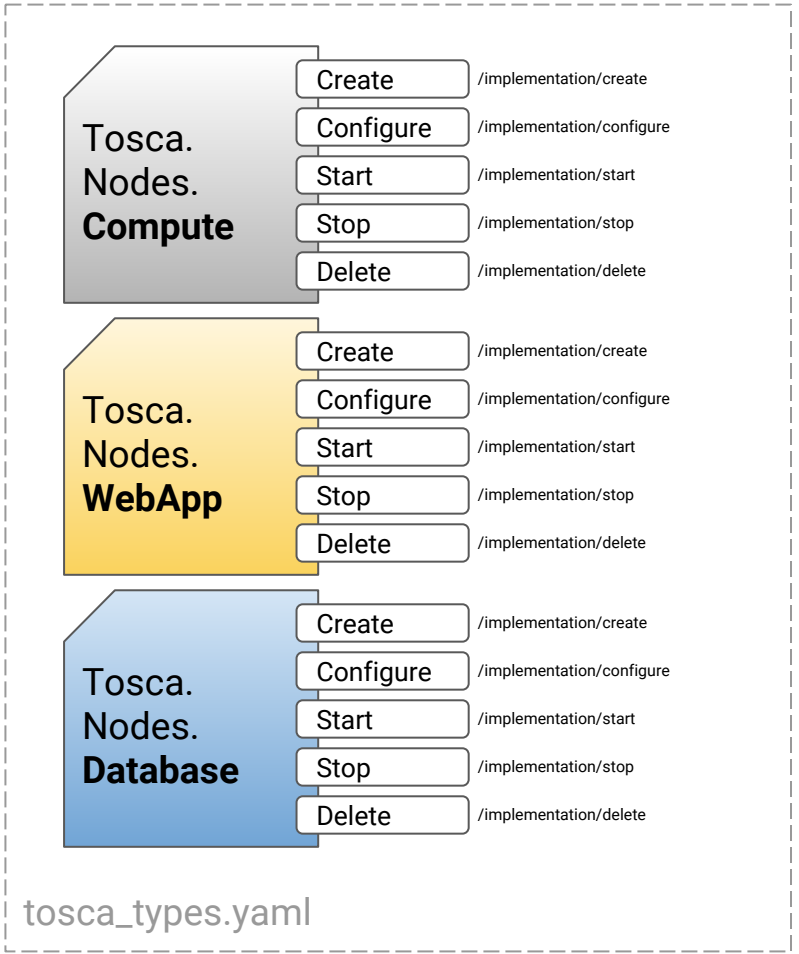
ARIA

Architecture

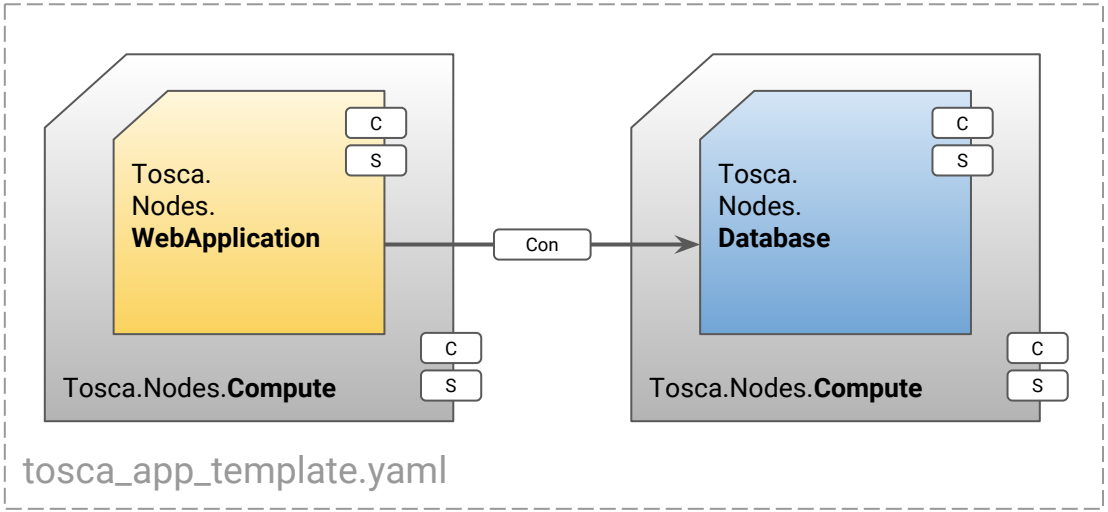
Declarative Model-Driven Orchestration

TOSCA Types

- Normative node types (Compute, Network, etc')
- Relationship types.
- Lifecycle operations and implementations
  (create, configure, start and others)
- ARIA supports custom types.

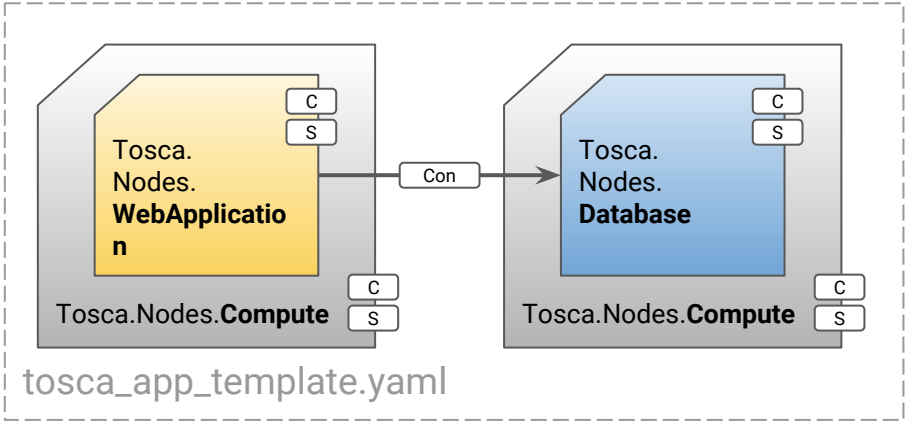


TOSCA Application Topology

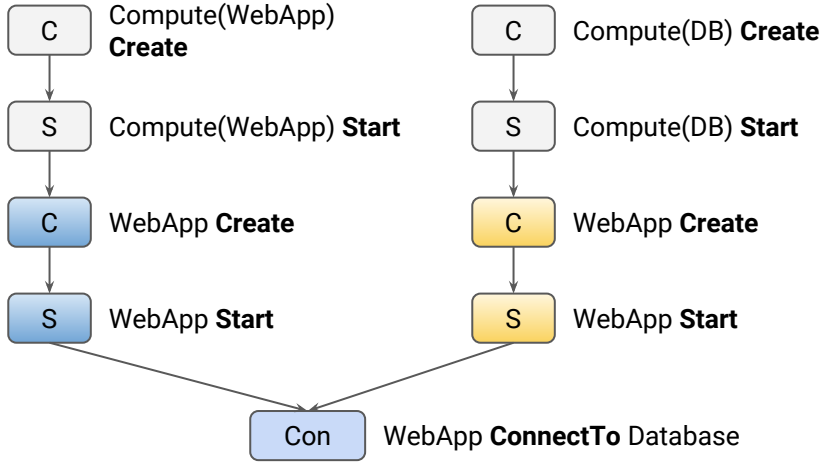


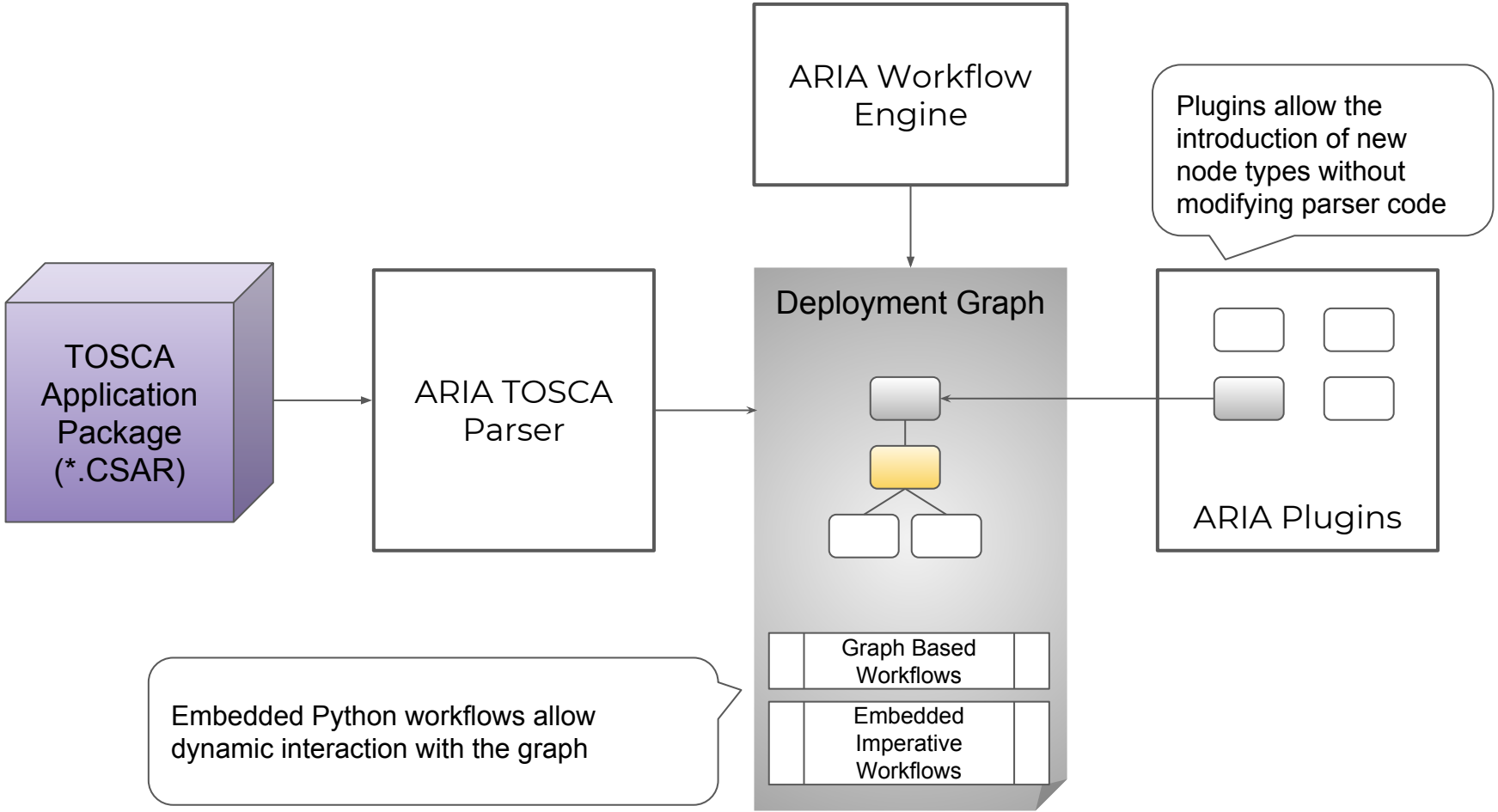
- TOSCA Template describes the topology of the application.
- Topology templates use TOSCA node types describing the nodes and their relationships using normative and custom types.
- Templates can also define implementations for lifecycle operations.

Declarative Model-Driven Orchestration

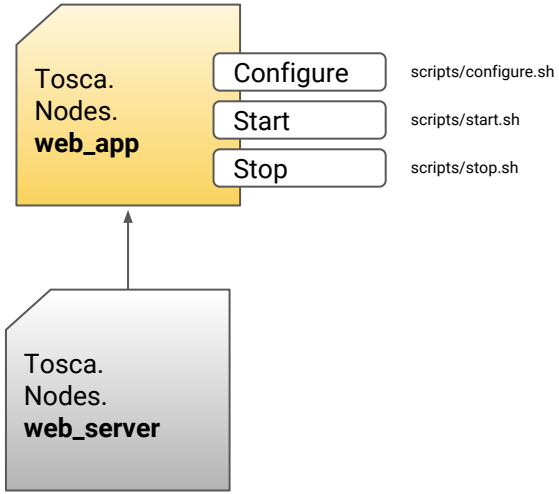


- ARIA parses application template
- Normalizes the topology
- Orchestration Graph is created
- A workflow mechanism loads the orchestration graph and executes operations of the graph
- **Deploy** and **Undeploy** are common, other day 2 operations such as Heal are supported as well.




ARIA Workflow Engine
Plugins allow the introduction of new node types without modifying parser code
TOSCA Application Package (*.CSAR)
ARIA TOSCA Parser
Deployment Graph
ARIA Plugins
Graph Based Workflows
Embedded Imperative Workflows
Embedded Python workflows allow dynamic interaction with the graph

Hello World Example



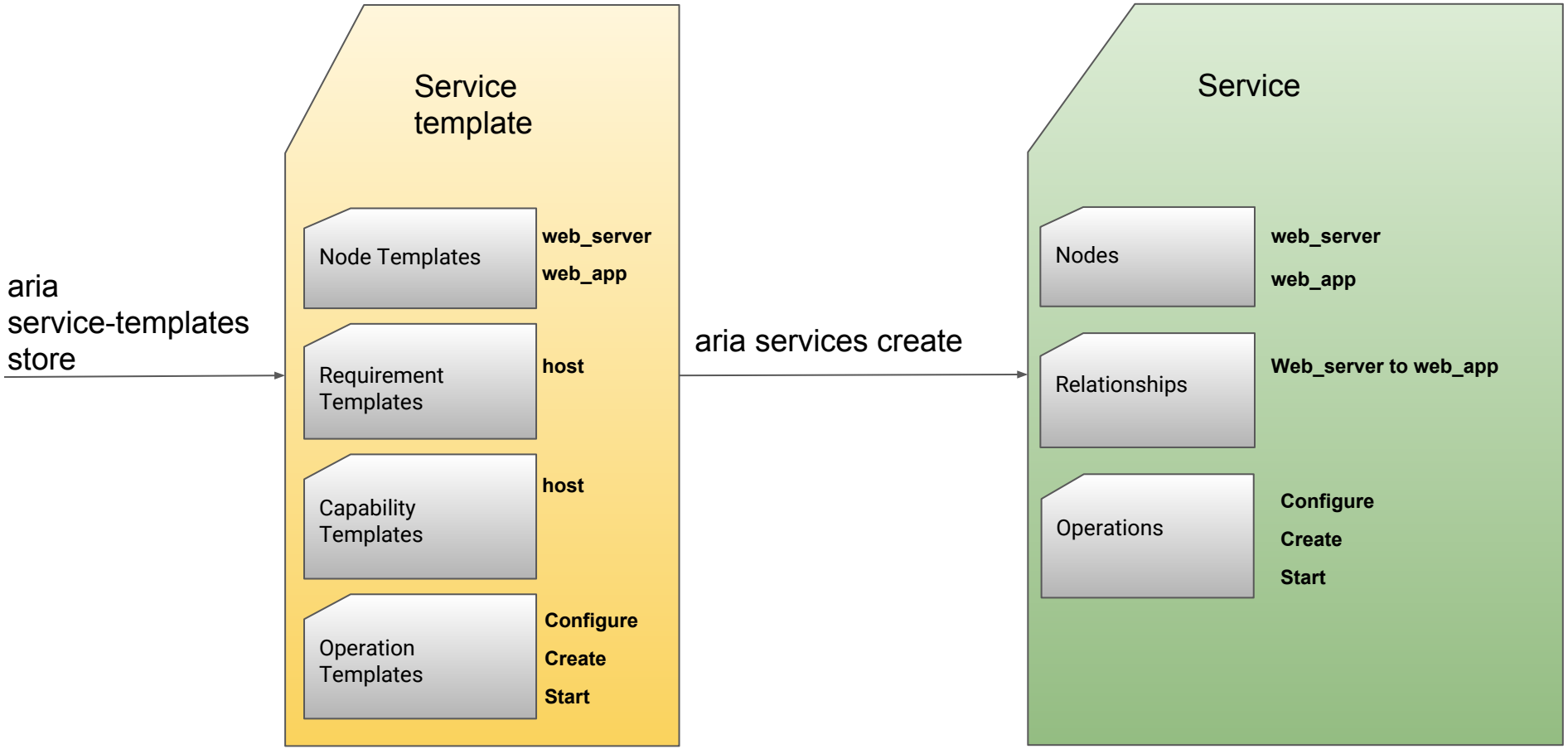

```
tosca_definitions_version: tosca_simple_yaml_1_0
node_types:
  web_server:
    derived_from: tosca.nodes.Root
    capabilities:
      host:
        type: tosca.capabilities.Container
  web_app:
    derived_from: tosca.nodes.WebApplication
    properties:
      port:
        type: integer
topology_template:
  node_templates:
    web_server:
      type: web_server
    web_app:
      type: web_app
      properties:
        port: 9090
      requirements:
        - host: web_server
      interfaces:
        Standard:
          configure: scripts/configure.sh
          start: scripts/start.sh
          stop: scripts/stop.sh
```

helloworld.yaml

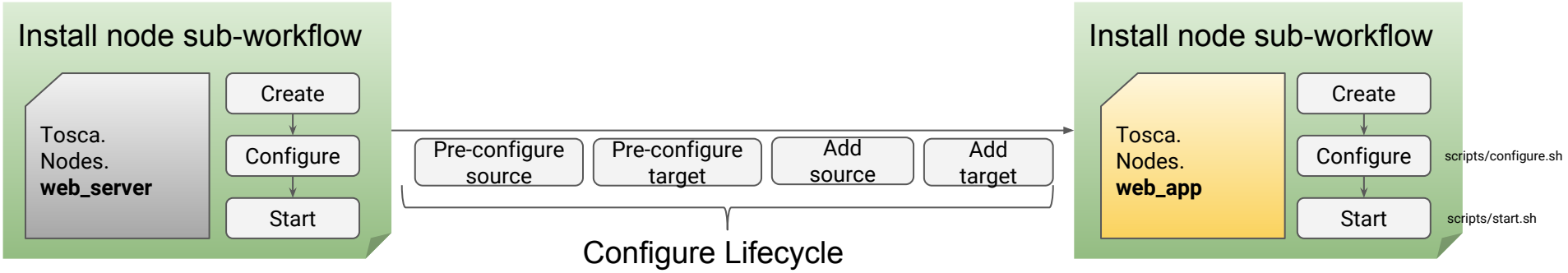
pip install git+https://github.com/apache/incubator-ariatosca.git
aria service-templates store **helloworld.yaml hello**
aria services create **hello** -t **hello**
aria executions start install -s **hello**

https://github.com/apache/incubator-ariatosca/tree/master/examples/hello-world

From topology template to a Service



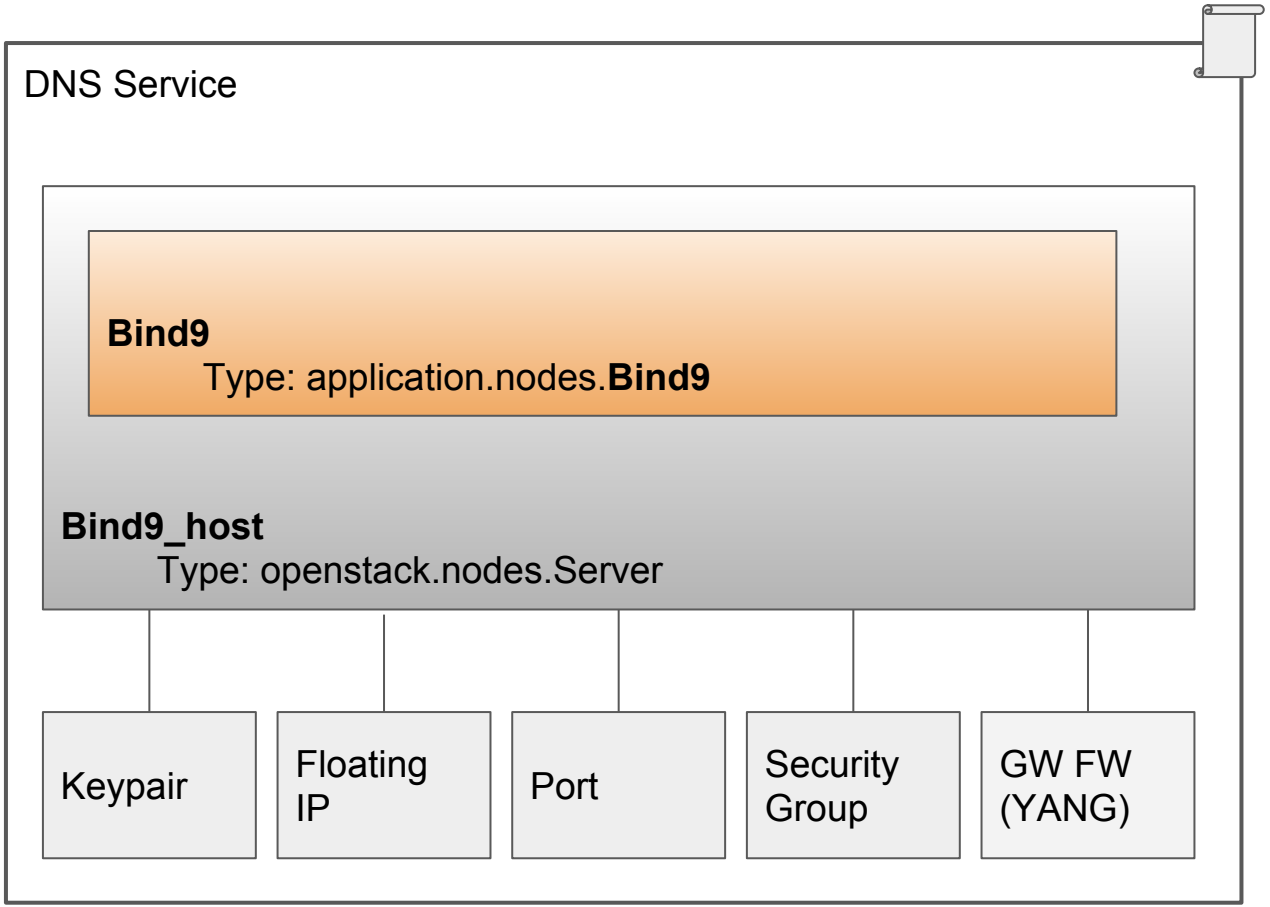
Install Workflow Breakdown



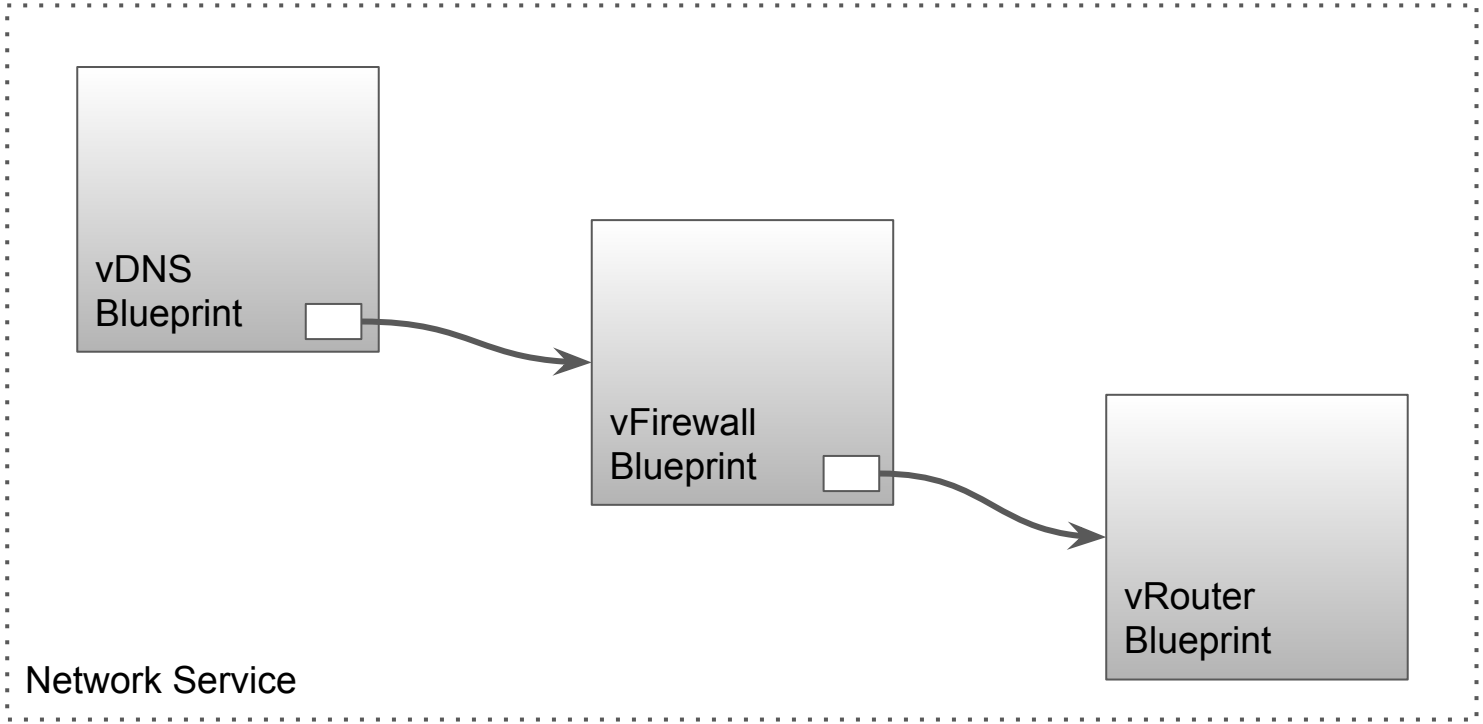

```python
@workflow
def install(ctx, graph):
    tasks_and_nodes = []
    for node in ctx.nodes:
        tasks_and_nodes.append((api_task.WorkflowTask(workflows.install_node, no
    graph.add_tasks([task for task, _ in tasks_and_nodes])
    workflows.create_node_task_dependencies(graph, tasks_and_nodes)
```

```python
@workflow(suffix_template='{node.name}')
def install_node(graph, node, **kwargs):
    # Create
    sequence = [task.create_task(node, NORMATIVE_STANDARD_INTERFACE, NORMATIVE_CREATE)]

    # Configure
    sequence += task.create_relationships_tasks(node,
                                                NORMATIVE_CONFIGURE_INTERFACE,
                                                NORMATIVE_PRE_CONFIGURE_SOURCE,
                                                NORMATIVE_PRE_CONFIGURE_TARGET)
    sequence.append(task.create_task(node, NORMATIVE_STANDARD_INTERFACE, NORMATIVE_CONFIGURE))
    sequence += task.create_relationships_tasks(node,
                                                NORMATIVE_CONFIGURE_INTERFACE,
                                                NORMATIVE_POST_CONFIGURE_SOURCE,
                                                NORMATIVE_POST_CONFIGURE_TARGET)
    # Start
    sequence += _create_start_tasks(node)

    graph.sequence(*sequence)
```

Modelling DNS Service with OpenStack



Service Function Chaining with TOSCA



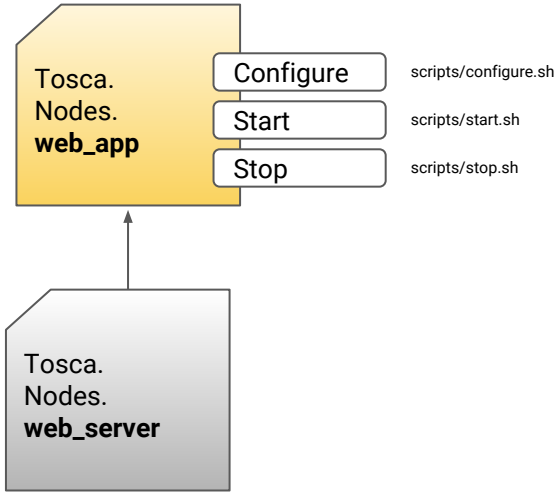
Join the community

www.ARIATOSCA.org

https://github.com/apache/incubator-ariatosca

dev@ariatosca.apache.org

Hello World Example



pip install git+https://github.com/apache/incubator-ariatosca.git
aria service-templates store **<helloworld.yaml> <service_name>**
aria services create **<service_name>** -t **<template_hello>**
aria executions start install -s **<service_name>**

```
tosca_definitions_version: tosca_simple_yaml_1_0
node_types:
  web_server:
    derived_from: tosca.nodes.Root
    capabilities:
      host:
        type: tosca.capabilities.Container
  web_app:
    derived_from: tosca.nodes.WebApplication
    properties:
      port:
        type: integer
topology_template:
  node_templates:
    web_server:
      type: web_server
    web_app:
      type: web_app
      properties:
        port: 9090
      requirements:
        - host: web_server
      interfaces:
        Standard:
          configure: scripts/configure.sh
          start: scripts/start.sh
          stop: scripts/stop.sh
```

helloworld.yaml

https://github.com/apache/incubator-ariatosca/tree/master/examples/hello-world