TOSCA Substitution Mappings

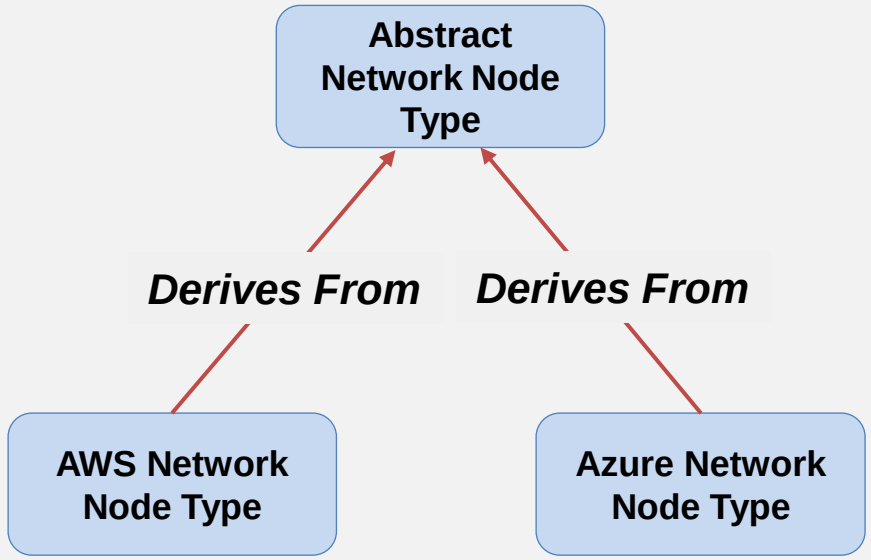
July 6, 2021

Abstraction in TOSCA

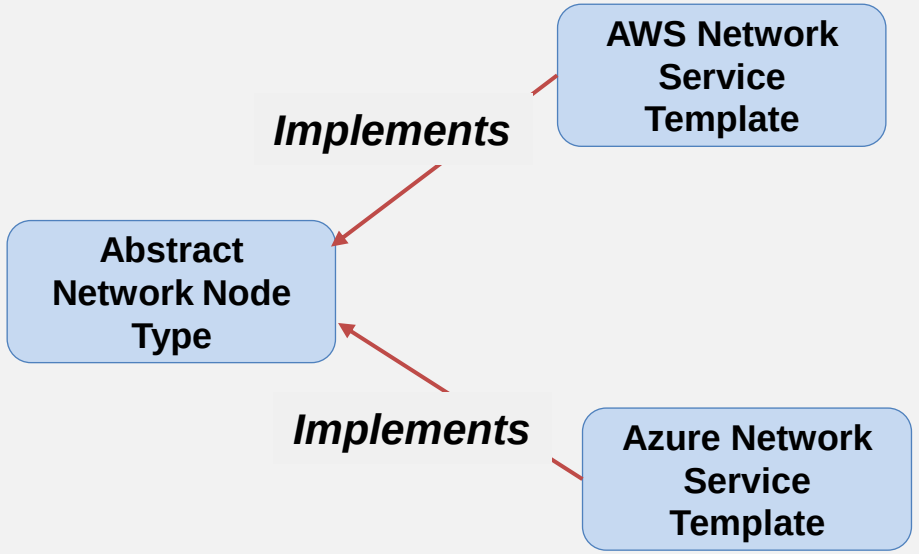
Goal of abstraction

- Avoid making technology/product decisions *at design time or service order time.*
- TOSCA patterns in support of abstraction

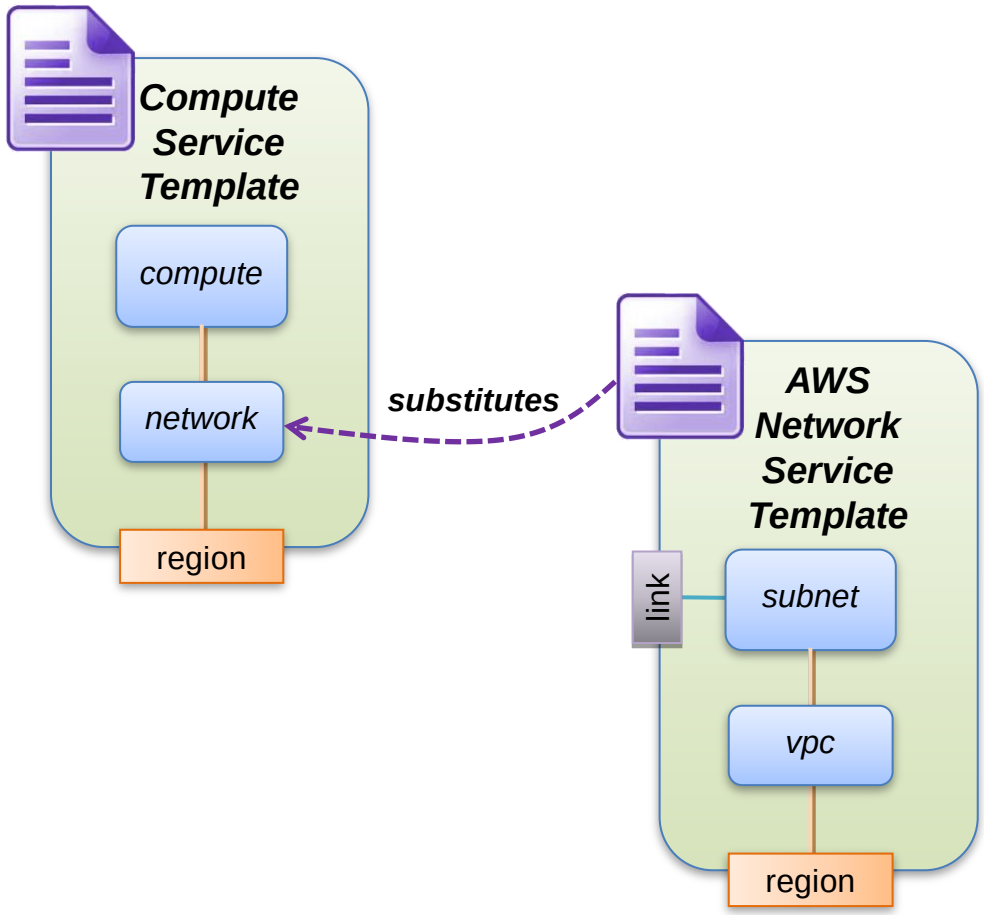
- **Abstract Base Class / Concrete Derived Classes**



- **Abstract Interface Class / Concrete Implementations**



Substitution Mapping: the TOSCA Interface/Implementation Pattern



Substituting Service Templates provide Implementation for Abstract Nodes

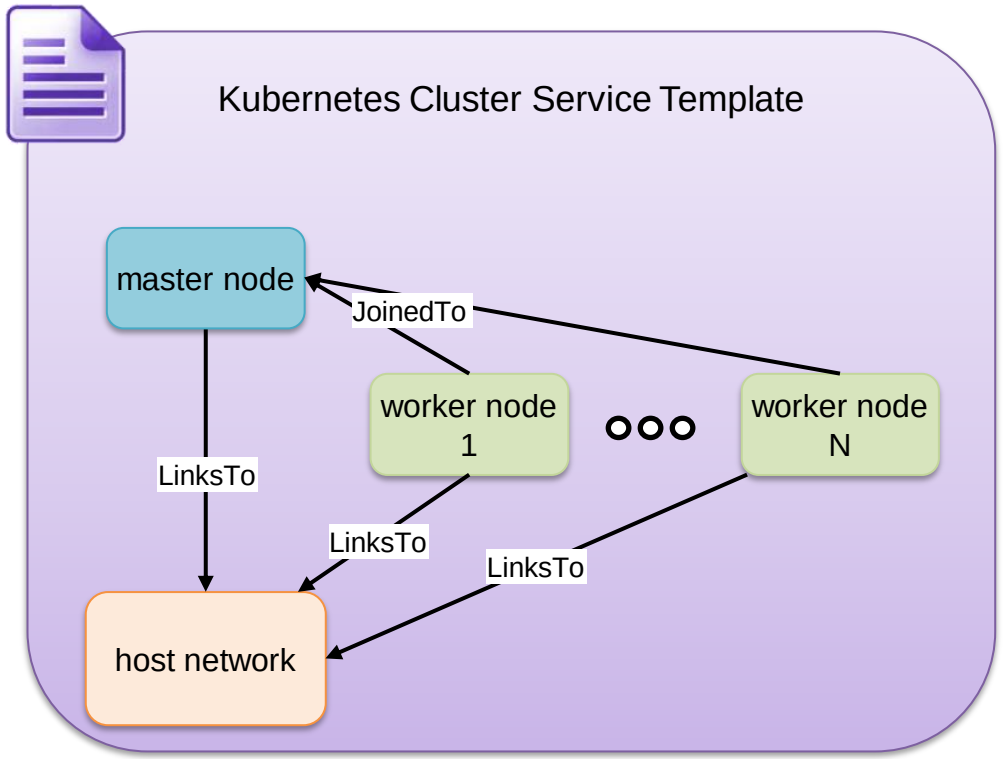
– Using substitution mapping created at service deployment time

Benefits of Interface/Implementation pattern

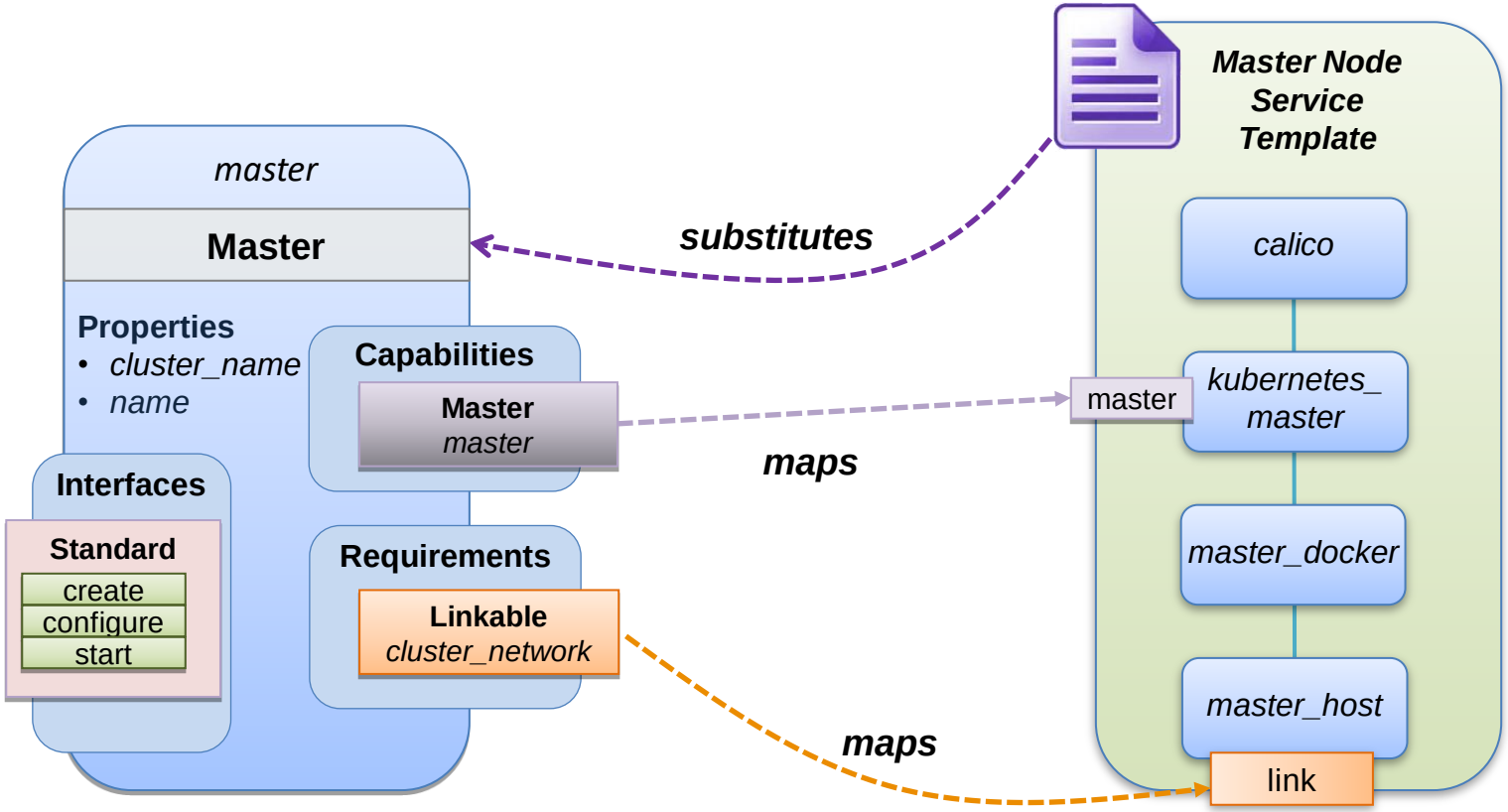
– *Declarative*: implementation is defined using service topology models
  - Rather than imperatively using code

– *Loose coupling*: allows changing implementations dynamically at runtime
  - E.g., deployment flavor in NFV

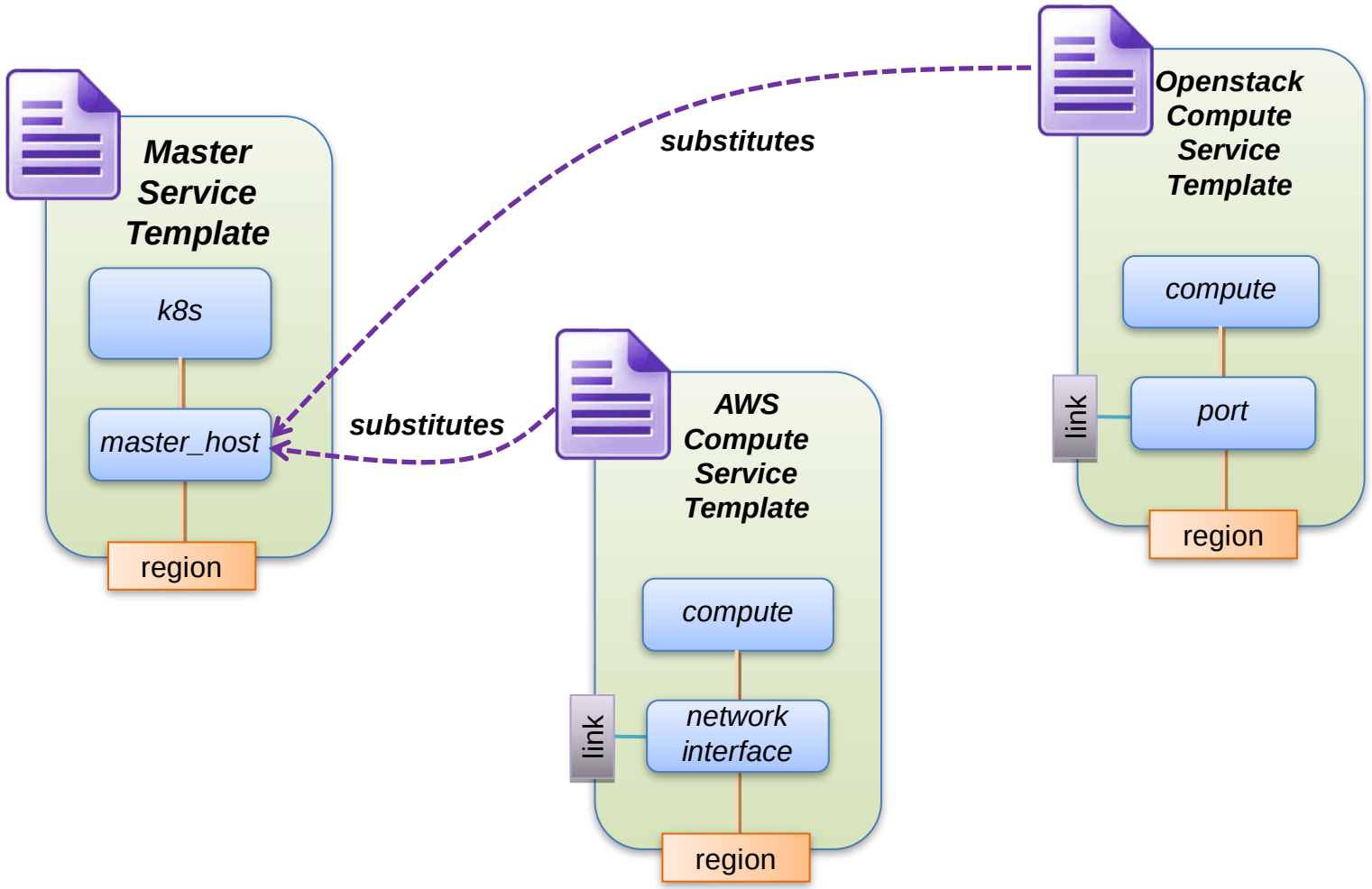
Example: Abstract Kubernetes Cluster Service Template



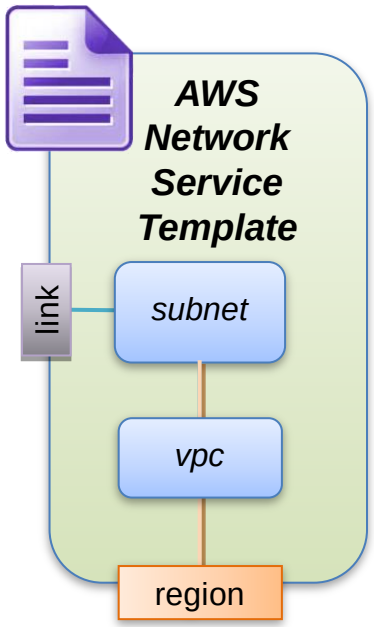
Substitution Mapping—Implementing Abstract Nodes



Substitution Mapping—Platform-Specific Implementations

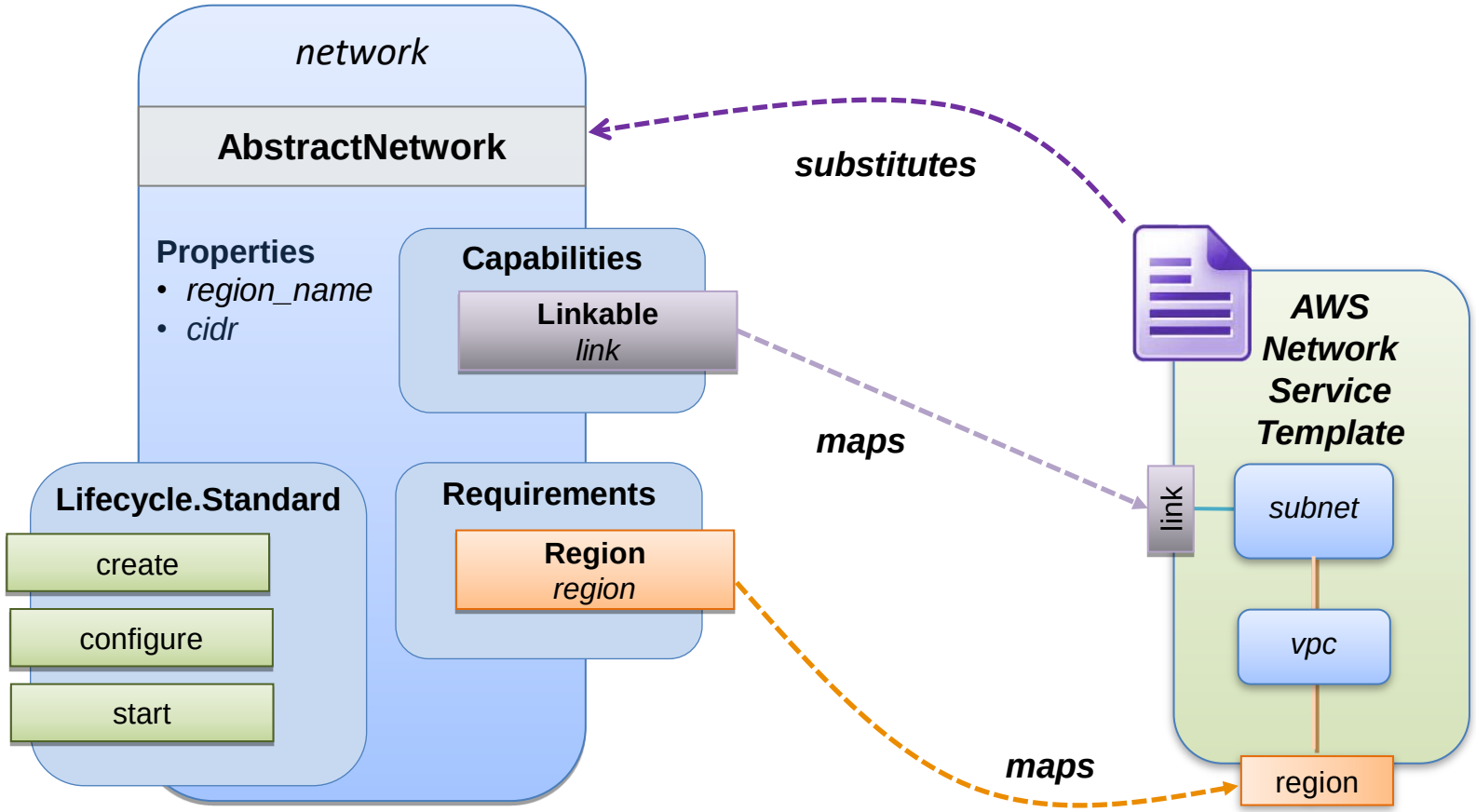


TOSCA Service Templates provide Implementations—Using Service Topology Graphs




```
tosca_definitions_version: tosca_simple_yaml_1_3
topology_template:
inputs:
    region_name:
      type: string
    cidr:
      type: string
  node_templates:
    subnet:
      type: aws:Subnet
      properties:
        cidr_block: { get_input: cidr }
      requirements:
        - vpc: vpc
    vpc:
      type: aws:VirtualPrivateCloud
      properties:
        region_name: { get_input: region_name }
```

Substitution Mapping—Stitching Substituting Template into Containing Topology



Elements of Substitution Mappings—Substituted Type

Primary goal of substitution mappings is to specify the **type of the nodes** they can **substitute**

```
topology_template:
 substitution_mappings:
    node_type: AbstractNetwork
    properties:
      region_name:  [ region_name ]
      cidr:         [ cidr ]
    requirements:
      region:       [ vpc, region ]
    capabilities:
      link:         [ subnet, link ]
    interfaces:
      Standard:
        create: deploy
        delete: undeploy
    substitution_filter:
      properties:
        - region_name: { equal: AWS Region }
```

Elements of Substitution Mappings—Properties and Attributes

Property mappings **map** properties of the abstract node onto inputs of the substituting template

Shows how ***information flows*** between the abstract node and its substituting topology during orchestration.

```
topology_template:
 substitution_mappings:
    node_type: AbstractNetwork
    properties:
      region_name:  [ region_name ]
      cidr:         [ cidr ]
    requirements:
      region:       [ vpc, region ]
    capabilities:
      link:         [ subnet, link ]
    interfaces:
      Standard:
        create: deploy
        delete: undeploy
    substitution_filter:
      properties:
        - region_name: { equal: AWS Region }
```

Elements of Substitution Mappings—Mapping Requirements and Capabilities

Requirement mappings **map** requirements and capabilities of the abstract node onto requirements and capabilities of nodes of the substituting template

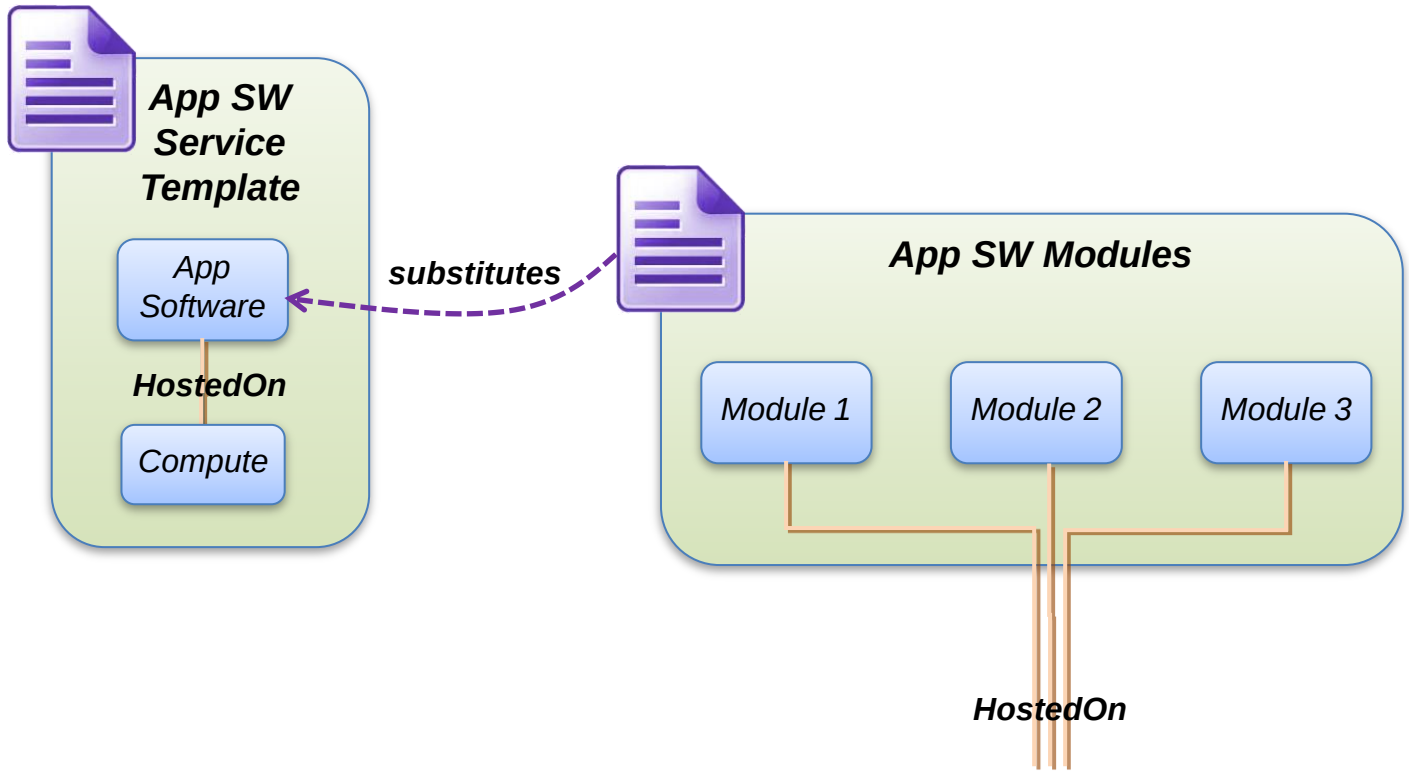
Shows how the substituting topology is ***stitched*** into the topology graph that contains the abstract (substituted) node.

```
topology_template:
 substitution_mappings:
    node_type: AbstractNetwork
    properties:
      region_name:  [ region_name ]
      cidr:         [ cidr ]
    requirements:
      region:       [ vpc, region ]
    capabilities:
      link:         [ subnet, link ]
    interfaces:
      Standard:
        create: deploy
        delete: undeploy
    substitution_filter:
      properties:
        - region_name: { equal: AWS Region }
```

Valid Requirement Mappings

Top-level node	Node in substituting template	Valid
Not assigned	not assigned	Should Not Be (bad substituting template design anyway)with new interpretation, this will be valid
fixed	not assigned	Noyes, if corresponds with type definition
dangling	Not assigned	No, yes if corresponds with type definition
not assigned	fixed	No
fixed	fixed	No
dangling	fixed	No
not assigned	dangling	Yes (of course if type occurrences [0,<any>])
fixed	dangling	Yes
dangling	dangling	Yes

Mapping to Multiple Requirements

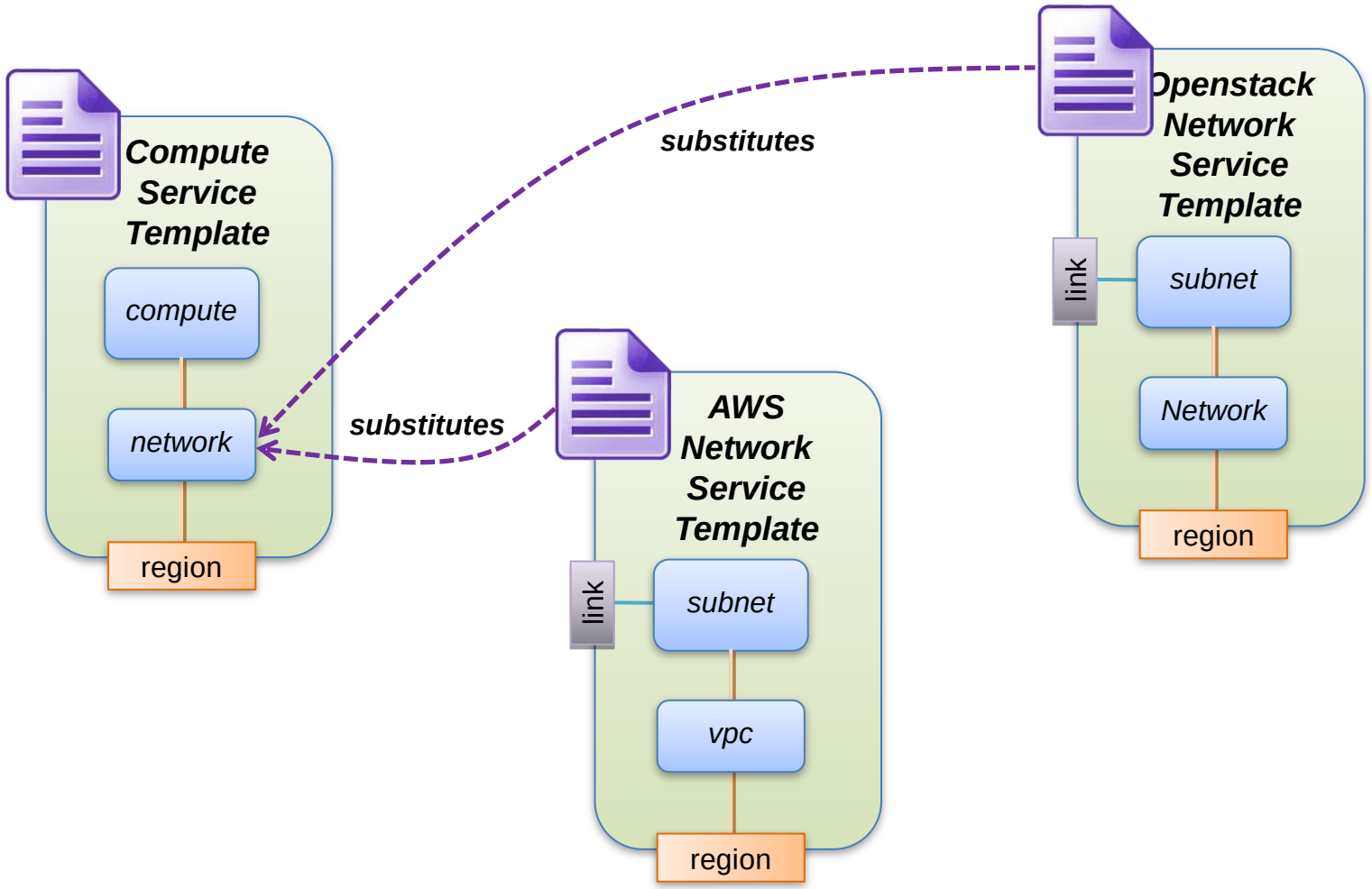


Elements of Substitution Mappings—Mapping Interface Operations

Interface mappings **map** interface operations of the abstract node onto workflows of the substituting template

```
topology_template:
 substitution_mappings:
    node_type: AbstractNetwork
    properties:
      region_name:  [ region_name ]
      cidr:         [ cidr ]
    requirements:
      region:       [ vpc, region ]
    capabilities:
      link:         [ subnet, link ]
    interfaces:
      Standard:
        create: deploy
        delete: undeploy
    substitution_filter:
      properties:
        - region_name: { equal: AWS Region }
```

Multiple Substitution Candidates—Controlling Candidate Selection



Elements of Substitution Mappings—Substitution Filter

To be substitutable by this template, property values of abstract node must **satisfy constraints** specified in the substitution_filter

Substitution filter **limits the number of abstract nodes** for which this service template is a valid substitution.

```
topology_template:
 substitution_mappings:
    node_type: AbstractNetwork
    properties:
      region_name:  [ region_name ]
      cidr:         [ cidr ]
    requirements:
      region:       [ vpc, region ]
    capabilities:
      link:         [ subnet, link ]
    interfaces:
      Standard:
        create: deploy
        delete: undeploy
    substitution_filter:
      properties:
        - region_name: { equal: AWS Region }
```

General Assumption

Substituting template must be valid TOSCA

- Can be deployed stand-alone
- i.e. without being used as a substituting template

Open Issues

Property Mappings

– Property mapping syntax is unnecessarily complex
  - Uses single-element lists to specify inputs
– No syntax for mapping properties of capabilities onto inputs
  - Which might be OK
  - See ***capability mapping*** section

Attribute Mappings

– Attribute mapping syntax is unnecessarily complex
  - Uses single-element lists to specify outputs

Topology Inputs

– Substituting topology may require domain or platform-specific inputs
– That cannot be retrieved from properties of the abstract node

Open Issues

Mapping Inputs and Outputs

– When to map properties and attributes of the abstract node onto inputs and outputs of the substituting topology?

– When to map operation inputs and outputs of the abstract node onto inputs and outputs of workflows of the substituting topology?

Multiple Occurrences of the Same Requirement

– How to map different occurrences of the same requirement of the abstract node onto different requirements of nodes in the substituting topology?

Capability Mapping

– When do properties of capabilities of the abstract node get propagated to properties of capabilities of nodes in the substituting topology?

– When do properties of capabilities of nodes in the substituting topology get propagated to properties of capabilities of the abstract node?

Open Issues

Notifications

– No syntax for mapping notifications of nodes in the substituting topology to notifications on the abstract node