Defining InterCloud Architecture (for Cloud based Infrastructure Services provisioned on-demand) and Cloud Security Infrastructure

Yuri Demchenko
SNE Group, University of Amsterdam

Cloud Federation Workshop, OGF32
16 July 2011, Salt Lake City
Outline

• General use cases for InterCloud Architecture
  – Provisioning Cloud based project oriented infrastructures on-demand and distributed virtualised applications mobility
• Existing standardisation and other initiatives
  – IEEE InterCloud Working Group(s)
  – DMTF OVF and VMware vApp
• Architectural Framework of the Cloud IaaS Provisioning Model (by UvA)
• Cloud Security Infrastructure and Dynamic Security Services Provisioning
  – Security Context Management
SNE Cloud Research Directions

(1) Generic Cloud IaaS Architecture, Release 1, 15 April 2011

- Infrastructure Services Modeling Framework (ISMF)
- Composable Services Architecture (CSA)
- Service Delivery Framework (SDF)

(2) InterCloud (OS/Middleware)
- Targeting for InterCloud BGP-like protocol
- Merging (1) and (2) under InterCloud Architecture
  - Network infrastructure provisioning as part of Cloud infrastructure provisioning

(3) Security Infrastructure for Cloud (dynamically provisioned)
- Dynamic Access Control Infrastructure (DACI)

- Following Cloud standardisation and contributing to NIST Cloud collaboration
General use cases for InterCloud Architecture

• Clouds are evolving as a common way of provisioning infrastructure services on-demand
  – In this way, Clouds add a new type of services in addition and on the top of currently existing network based and distributed services
  – Using real life analogy “like moving house”

• InterCloud Architecture (ICA) provide a framework to support provisioning Cloud based project oriented infrastructures on-demand and distributed virtualised applications mobility
  – Hybrid Cloud/Grid e-Science collaborative environment
  – Educational Lab deployment in Clouds

• Other use cases to be defined
Use case 1: Cloud based e-Science infrastructure

Project based Collaborating user groups located in remote campuses on data intensive projects requiring high performance computing and rich visualisation.

Cloud infrastructure provisioned on demand

Grid based core eScience Infrastructure including data intensive scientific instrument

Campus infrastructure including visualisation tools

User Group A

User Group B

Control & Monitoring Plane

Experimental Data

Specialist Data Processing

Processed Data

CloudSE T1

Grid CE Data Filtering

Grid Storage T1

Sc. Instrument (Manufactrg)

Grid Storage T0

Campus A

Campus B

Visualisation

User

CE
Use case 2: Educational Lab (mobility)

- Educational lab is created for a specific course in one university
  - A course is computing intensive and has periodicity of one semester
    - The required infrastructure is expensive and is deployed on Cloud (generally multiple)
    - First installation requires significant efforts that need to preserved
- Between periodic course runs the Lab will be dormant or should be suspended and resumed for the next term
  - Used/required Cloud resources may change/evolve
- The Lab may need to be moved to another university with different campus network installation and available Cloud providers
  - Requires Cloud services standardisation and interoperability
InterCloud: Related standardisation activity

- IEEE - WGs on InterCloud issues and Cloud Profiles
  - IEEE ICWG/2302 WG - Intercloud WG (ICWG) Working Group
    http://standards.ieee.org/develop/wg/ICWG-2302_WG.html
  - CPWG/2301 WG - Cloud Profiles WG (CPWG) Working Group
    http://standards.ieee.org/develop/wg/CPWG-2301_WG.html

- DMTF OVF (http://www.dmtf.org/standards/ovf)
  - Supported by VMware OVF Tool 2.0

- VMware vApp
vApp by VMware

- vApp container for distributed multi-VM solution
  - Decouples application from the deployment platform
  - Built on top of OVF2.0
  - Implemented as a vApprun package

- Concept of a vService dependency is used to decouple a vApp from infrastructure services and to support mobility between cloud providers

vApp: a standards-based container for cloud providers, by R.Schmidt, S.Grarup
[http://portal.acm.org/citation.cfm?id=1899943](http://portal.acm.org/citation.cfm?id=1899943)
Defining InterCloud Architecture

• The prospective InterCloud Architecture should allow interoperability and integration of existing models and Cloud providers frameworks
  – Should be superseded to Cloud Federation
• Be compatible and provide multi-layer integration of existing Cloud service models – IaaS, PaaS, SaaS and Apps clouds
• Presumably following the same architecture patterns as Internet and Grid/OGSA
  – Provide functionalities for creating VO based infrastructures
Current relation between Cloud services models

- Cloud service models IaaS, PaaS, SaaS use proprietary Physical Platform and Resources Adaptation Layer

Diagram:

- Customers & Applications
  - Cloud SaaS (Apps)
    - Cloud PaaS (OS, mw)
      - Cloud IaaS (VM MgmtS)
  - API (Data, C&MP)
    - User and Application API (Data, C&MP)

- Physical Platform and Resources Adaptation Layer (PPR Adaptation)
- Computer Platform
Prospective InterCloud Architecture

- Standardisation API’s between different Cloud service models
- Cloud/ICA layered API
  - For application data communication
  - For Control and Management

Customers & Applications

Cloud SaaS (Apps)

Cloud PaaS (OS, mw)

Cloud IaaS (VM MgmtS)

Physical Platform and Resources Adaptation Layer

Computer Platform
Defining InterCloud Architecture API’s

• InterCloud Architecture (ICA) should address interoperability of different Cloud service platforms and multi-cloud integration, including with legacy campus infrastructure

• Define InterCloud protocols and API’s stack
  – VI-API – IaaS API
  – P-API – PaaS API
  – SA-API – Software (and applications) API
    – *OCCI can be a base for defining most of APIs*

• Depending on service model, some API’s may be run by providers and some by customers/users
Published as SNE Technical Report

• Includes the following main components
  – Infrastructure Services Modeling Framework (ISMF)
  – Composable Services Architecture (CSA)
  – Service Delivery Framework (SDF)

• Additional components (orthogonal)
  – Cloud Security Infrastructure
  – Control and Management Plane
Use case 1: Cloud based e-Science infrastructure

Project based Collaborating user groups located in remote campuses on data intensive projects requiring high performance computing and rich visualisation.

Cloud infrastructure provisioned on demand.

Campus infrastructure including visualisation tools.

Grid based core eScience Infrastructure including data intensive scientific instrument.
Abstract Use Case - IaaS General Model

Cloud Federation @OGF32 - 16 July 2011, Salt Lake City
The proposed framework should support on-demand infrastructure services provisioning and operation

- **Composable Services Architecture (CSA)** that intends to provide a conceptual and methodological framework for developing dynamically configurable virtualised infrastructure services
- **Service Delivery Framework (SDF)** that provides a basis for defining the whole composable services life cycle management and supporting infrastructure services
- **Infrastructure Services Modeling Framework (ISMF)** that provides a basis for the infrastructure resources virtualisation and management, including description, discovery, modeling, composition and monitoring
- (Additionally) **Service Control and Management Plane/Framework** may be defined as combination of management functionality in all 3 components
- (Additionally) **Security services/infrastructure** have a dual role:
  - Virtual Security Infrastructure - provisioned as a part of virtualised infrastructure
  - Support normal/secure operation of the whole provisioning framework
Virtual Infrastructure Composition and Management (VICM) Layer Operation

• Main actors involved into provisioning process
  – Physical Infrastructure Provider (PIP)
  – Virtual Infrastructure Provider (VIP)
  – Virtual Infrastructure Operator (VIO)

• Virtual Infrastructure Composition and Management (VICM) layer includes
  – VICM middleware - defined as CSA
  – Logical Abstraction Layer and the VI/VR Adaptation Layer facing correspondingly lower PIP and upper Application layer.

• The infrastructure provisioning process is defined by the Service Delivery Framework (SDF)

• VICM redefines Logical Infrastructure Composition Layer (LICL) proposed by GEYSERS project
  – Basic functionality is implemented as GEMBus/CSA
ISMF – Virtual Resource Lifecycle

- Planning
- Composition
- Reservation

LR2 -> VR
VI Deployment

(LR0) -> LR2
Planning
Composition
Reservation

PR-LR1
Config & Instantiation

LR0
Re-usable (Published)
PRs

Registered
PRs

Composed
LRs

Deployed
VRs

Network Segment
Toplogy Pool

Virtual Resource Lifecycle

Physical Resource

Logical Resource

Virtual Infrastructure
Virtual Resource lifecycle – defines relations between different resource presentations along the provisioning process

Physical Resource information is published by PIP to the Registry service serving VICM and VIP
- Logical Resource representing PR includes also properties that define possible (topological) operations on the PR, such as e.g. partitioning or aggregation.

Published LR information presented in the commonly adopted form (using common data or semantic model) is then used by VICM/VIP composition service to create requested infrastructure as combination of (instantiated) Virtual Resources and interconnecting them with the available network infrastructure

Network infrastructure can be composed of a few network segments (from the network topology pool) run by different network providers.

Composed LRs are deployed as VRI/VI to VIP/VIO and as virtualised/instantiated PR-LR to PIP

Resource/service description format considered
- NDL/NML (Network Description Language / Network Markup Language at OGF)
- Compatibility with VXDL infrastructure service request format by INRIA
Composable Services Architecture (CSA)

• Defined as middleware for on-demand provisioned Composable Services
• Proposed in the GEANT3 JRA3 Composable Services project
• Implemented as GEMBus (GEANT Multidomain Bus)
Composable Services Architecture – Version 0.13

Applications and User Terminals

Proxy (adaptors/containers) – Composed/Virtualised Services and Resources

Control & Management Plane
(Orchestration)

Composable Services Middleware (GEMBus)

Composition Layer/Serv (Reservation SLA Negotiation)

Logical Abstraction Layer for Component Services and Resources

Proxy (adaptors/containers) – Component Services and Resources

MD SLA
Registry
Logging
Security

Storage Resources
Compute Resources
Network Infrastructure

Component Services & Resources

Composable Services lifecycle/provisioning stages
(1) Request
(2) Composition/Reservation
(3) Deployment
(4) Operation
(5) Decommissioning

Control/ Mngnt Links
Data Links
Composable Services Lifecycle/Provisioning Workflow

- **Main stages/phases**
  - Service Request (including SLA negotiation)
  - Composition/Reservation (aka design)
  - Deployment, including Registration/Synchronisation
  - Operation (including Monitoring and SLA enforcement)
  - Decommissioning (including Dynamic Security Associations destroying/recycling)

- **Additional stages**
  - Re-Composition should address incremental infrastructure changes
  - Recovery/Migration can use SL-MD to initiate resources re-synchronisation but may require re-composition

- **The whole workflow is supported by the Service Lifecycle Metadata Service (SL MD)**

- **Provisioning session provides a framework for services context and security context management**
Part 2: Cloud Security Infrastructure

- Provides a framework for dynamically provisioned Cloud security services and infrastructure
Cloud Security – Issues and problem environment

- Virtualised services
- On-demand/dynamic provisioning
- Multi-tenant/multi-user
- Multi-domain
- Uncontrolled execution and data storage environment
  - Data protection
    - Trusted Computing Platform Architecture (TCPA)
    - Promising homomorphic/elastic encryption
- Bootstrapping virtualised security infrastructure and virtualisation platform
- Integration with legacy security services/infrastructure of the providers
- Integration with the providers business workflow
Current Cloud Security Model

• SLA based security model
  – SLA between provider and user defines the provider responsibility and guarantees
    • Actually no provider responsibility
  – Providers undergo certification
  – Standard business model
• Using VPN and SSH keys generated for user infrastructure/VMs
  – Works for single Cloud provider
• Has inherited key management problems
• Not scalable
• Not easy integration with legacy physical resources
• Simple access control, however can be installed by user
• Trade-off between simplicity and manageability
New Relations in Cloud services provisioning

• Old model: Provider – User
• New relations with infrastructure services provisioning
  – Provider – Customer (University) – User (End user - Student)
  – Provider – Operator/Broker - Customer – User
• Rich (flexible and complex) ownership relations

• Two research projects are underway
  – Trust management in Cloud based infrastructure services provisioning
    • RORA model – Resource – Ownership – Roles - Actors
  – Virtual Infrastructure bootstrapping protocol
Specific SSLM stages and mechanisms to ensure consistency of the security context management

- **Security Service Request** that initiates creation of the dynamic security association and may use SLA security context.
- **Reservation Session Binding** with GRI (also a part of general SDF/SLM) that provides support for complex reservation process including required access control and policy enforcement.
- **Registration & Synchronisation** stage (as part Deployment stage) that allows binding the local resource or hosting platform run-time process ID to the GRI as a provisioning session ID. Specifically targets possible scenarios with the provisioned services migration or restoration.
Security Context Management in VI/IaaS

SDF Stages

User/Applicatn

VIO

VIP

VIP DACI/AAI

PIPs

PIP AAI

SLA Negotiation

Planning, Reservation

Deployment, Activation

Operation

Decommissioning

VI/VR Request (TA0)

Generate GRI

VI reservation request
(including VIO TA1)

Return GRI

Generate a VI-GRI

AuthZ request (VI-GRI)

VR reservation request
(including VIP TA2)

Reservation confirmation with PR-LRI of committed PR

Mapping from the returned PR-LRI to VR-GRI for VIP

Generate a VI-GRI

Return GRI

Reserve complex resource at PIP(s) in daisy-chain mode or concurrent mode

SDF Stages

User/Applicatn

VIO

VIP

VIP DACI/AAI

PIPs

PIP AAI

SLA Negotiation

Planning, Reservation

Deployment, Activation

Operation

Decommissioning

VI/VR Request (TA0)

Generate GRI

VI reservation request
(including VIO TA1)

Return GRI

Generate a VI-GRI

AuthZ request (VI-GRI)

VR reservation request
(including VIP TA2)

Reservation confirmation with PR-LRI of committed PR

Mapping from the returned PR-LRI to VR-GRI for VIP

Generate a VI-GRI

Return GRI

Reserve complex resource at PIP(s) in daisy-chain mode or concurrent mode
Additional Information

• GEYSERS project AAI
• Using AuthZ tickets and tokens for access control and signaling
Authentication and Authorization Infrastructure (AAI) functionalities

- Access control: interfaces, VI provisioning service
  - Authentication – standard implementation
  - Authorization – primary focus
  - Identity management – to support multi-domain attributes management

- Security context management for VI provisioning service
  - Cross-layer/multi-layer
  - Inter-domain/dynamic security associations
  - Lifecycle and provisioning session security context

- Dynamic access control services/infrastructure
  - Security Services Lifecycle Management (SSLM)
  - Dynamic security/trust associations management
Basic CSSI services
- Data encryption
- Digital signature
- Authentication
- Authorization
- Policy management
- Security session and context management

CSSI – Common Security Services Interface
AAI in GEYSERS (3)
Multi-domain and Multi-layer Environment

Must provide AuthN assertion and support GRI and VI SecCtx

VIP/VIOSecurity domain

Upper-LICL:
- AuthN Service
- TVS
- AuthZ Service

Lower-LICL:
- AuthN Service
- TVS
- AuthZ Service

SML:
- AuthN Service
- TVS

NCP:
- AuthN Service
- TVS
- AuthZ Service

PHY1:
- AuthN Service
- TVS
- AuthZ Service

PHY2:
- AuthN Service
- TVS
- AuthZ Service

PIP Security domains
Access Token and Pilot Token Types

- **AType 0** – Simple access token (refers to the reserved resources context)
- **AType 1** – Access token containing Obligations collected from previous domains
- **PType 0** – Container for GRI only
- **PType 1** – Container for communicating the GRI during the reservation stage
  - Contains the mandatory SessionId=GRI attribute and an optional Condition element
- **PType 2** – Origin/requestor authenticating token
  - TokenValue element contains a value that can be used as the authentication value for the token origin
  - TokenValue may be calculated of the (GRI, IssuerId, TokenId) by applying e.g. HMAC function with the requestor’s symmetric or private key.
- **PType 3** – Extends Type 2 with the Domains element that allows collecting domains security context information when passing multiple domains during the reservation process
  - Domains’ context may include the previous token and the domain’s trust anchor or public key
- **PType 4** – Used at the deployment stage and can communicate between domains security context information about all participating in the provisioned lightpath or network infrastructure resources
  - Can be used for programming/setting up a TVS infrastructure for consistent access control tokens processing at the resource access stage
General XML Token Format – Access Token and Pilot Token

Required functionality to support multidomain provisioning scenarios
- Allows easy mapping to SAML and XACML related elements

Allows multiple Attributes format (semantics, namespaces)

Establish and maintain Trust relations between domains
- Including Delegation

Ensure Integrity of the AuthZ decision
- Keeps AuthN/AuthZ context
- Allow Obligated Decisions (e.g. XACML)
XML Token Example – Access Token Type 0

- `<AAA:AuthzToken xmlns:AAA="http://www.aaauthreach.org/ns/#AAA" Issuer="urn:aaa:gaaapi:token:TVS" type="access-type0"
  SessionId="a9bcf23e70dc0a0cd992bd24e37404c9e1709afb"
  TokenId="d1384ab54bd464d95549ee65cb172eb7">
  
  - `<AAA:TokenValue>ebd93120d4337bc3b959b2053e25ca5271a1c17e</AAA:TokenValue>
  
  - `<AAA:Conditions NotBefore="2007-08-12T16:00:29.593Z"
    NotOnOrAfter="2007-08-13T16:00:29.593Z"/>
  
  - `</AAA:AuthzToken>`

- where

  SessionId = GRI (Global Reservation Id)
  TokenId – unique identifier (serving for logging and accountability)
  TokenValue – generated securely from (GRI, TokenId, DomainId, TokenKey), or
  AuthzTicket (digital SignatureValue)

  - The element `<TokenValue>` and attributes SessionId and TokenId are mandatory, and
  the element `<Conditions>` and attributes Issuer, NotBefore, NotOnOrAfter are optional
  - Binary token contains just two values – TokenValue and GRI
Chaining Pilot Tokens in multidomain/multiprovider signalling

- Pilot Token type 3

```
<AAA:AuthzToken xmlns:AAA="http://www.aaauthreach.org/ns/AAA"
  Issuer=http://testbed.ist-phosphorus.eu/uva/AAA/TVS/tokenpilot
  SessionId="740b241e711ece3b128c97f990c282adcbf476bb"
  TokenId="dc58b505f9690692f7a6312912d0fb4c" type="pilot-type3">
  <AAA:TokenValue>190a3c1554a500e912ea75a367c822c09eceaa2f</AAA:TokenValue>
  <AAA:DomainsContext>
    <AAA:Domain domainId="http://testbed.ist-phosphorus.eu/viola">
        SessionId="2515ab7803a86397f3d60c670d199010aa96cb51"
        TokenId="c4a2f5f70346f6d2a2244fecbcdd244">
          <AAA:TokenValue>dee1c29719b9098b361cab4cfd086700ca2f414</AAA:TokenValue>
      </AAA:AuthzToken>
      <AAA:KeyInfo>http://testbed.ist-phosphorus.eu/viola/_public_key_</AAA:KeyInfo>
    </AAA:Domain>
  </AAA:DomainsContext>
</AAA:AuthzToken>
```