LECTURE NOTES ON

CLOUD COMPUTING

III B. Tech II semester (JNTUA-R15)

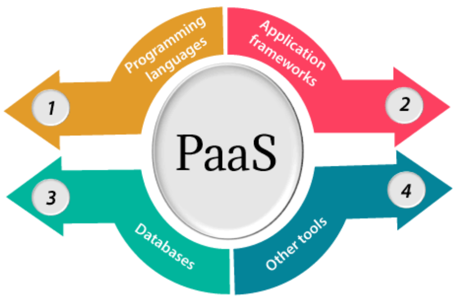
Mrs. S.Sunitha,
Assistant Professor
Department of CSE
G.Pullaiah college of Engineering and Technology
Kurnool.

Unit 3

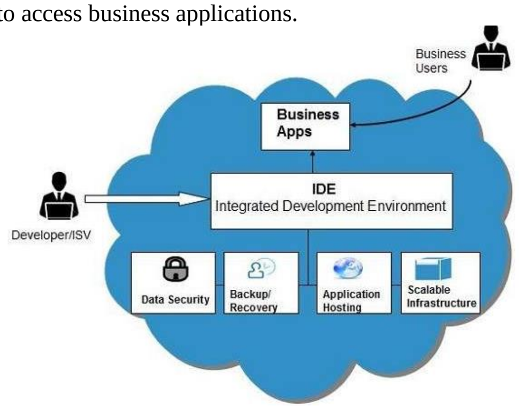
Platform as a Service

Platform as a Service (PaaS) provides a runtime environment. It allows programmers to easily create, test, run, and deploy web applications. You can purchase these applications from a cloud service provider on a pay-as-per use basis and access them using the Internet connection. In PaaS, back end scalability is managed by the cloud service provider, so end-users do not need to worry about managing the infrastructure. PaaS is a cloud delivery model for application composed of services managed by third party. In this computing model , providers rent developers everything they need to built an application: development tools , infrastructure and operating system. An internet connection is all that physically required to access Paas, making it possible for developers to build an entire application in a web browser.

PaaS includes infrastructure (servers, storage, and networking) and platform (middleware, development tools, database management systems, business intelligence, and more) to support the web application life cycle.
Example: Google App Engine, Force.com, Joyent, Azure.
PaaS providers provide the Programming languages, Application frameworks, Databases, and Other tools:



1. Programming languages
PaaS providers provide various programming languages for the developers to develop the applications. Some popular programming languages provided by PaaS providers are Java, PHP, Ruby, Perl, and Go.
2. Application frameworks
PaaS providers provide application frameworks to easily understand the application development. Some popular application frameworks provided by PaaS providers are Node.js, Drupal, Joomla, WordPress, Spring, Play, Rack, and Zend.
3. Databases
PaaS providers provide various databases such as ClearDB, PostgreSQL, MongoDB, and Redis to communicate with the applications.
4. Other tools
PaaS providers provide various other tools that are required to develop, test, and deploy the applications.
The following diagram shows how PaaS offers an API and development tools to the developers and how it helps the end user to access business applications.



Characteristics

Here are the characteristics of PaaS service model:

- PaaS offers **browser based development environment.** It allows the developer to create database and edit the application code either via Application Programming Interface or point-and-click tools. Online integrated development environment also known as web IDE or Cloud IDE ia a browser based integrated development environment. An IDE does not usually contain same features as traditional desktop IDE. IDE features a source code editor with syntax highlighting.
- PaaS provides **built-in security, scalability,** and **web service interfaces.** The cloud provided application development platform provides greater run and manage business applications.
- PaaS provides built-in tools for defining **workflow, approval processes,** and business rules. PaaS takes care of all the system administration details of setting up servers and virtual machines installing runtimes, libraries, middleware, configuring build and testing tools. The work flow in Paas is as simple as coding in the IDE and then pushing the code using tools like Git and seeing the changes immediately.
- It is easy to integrate PaaS with other applications on the same platform. Integration of application is a form of systems integration business delivered as a cloud computing service that addresses data, process, SOA and application integration. Cloud provided application development platform provides greater flexibility, as with other cloud services such as IaaS and public cloud PaaS, the customer controls software deployment while the PaaS services can also include web service integration.
- PaaS also provides web services interfaces that allow us to connect the applications outside the platform.

Advantages of PaaS

There are the following advantages of PaaS -

1) Simplified Development

PaaS allows developers to focus on development and innovation without worrying about infrastructure management.

2) Lower risk

No need for up-front investment in hardware and software. Developers only need a PC and an internet connection to start building applications.

3) Prebuilt business functionality

Some PaaS vendors also provide already defined business functionality so that users can avoid building everything from very scratch and hence can directly start the projects only.

4) Instant community

PaaS vendors frequently provide online communities where the developer can get the ideas to share experiences and seek advice from others.

5) Scalability

Applications deployed can scale from one to thousands of users without any changes to the applications.

Disadvantages of PaaS cloud computing layer

1) Vendor lock-in

One has to write the applications according to the platform provided by the PaaS vendor, so the migration of an application to another PaaS vendor would be a problem.

2) Data Privacy

Corporate data, whether it can be critical or not, will be private, so if it is not located within the walls of the company, there can be a risk in terms of privacy of data.

3) Integration with the rest of the systems applications

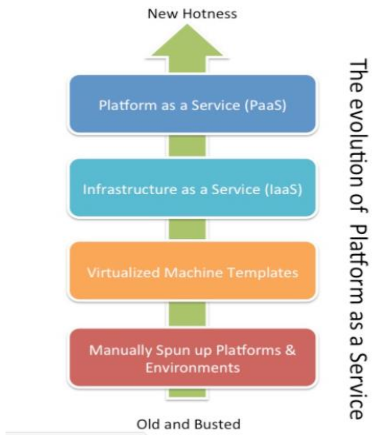
It may happen that some applications are local, and some are in the cloud. So there will be chances of increased complexity when we want to use data which in the cloud with the local data.

Architecture of PaaS

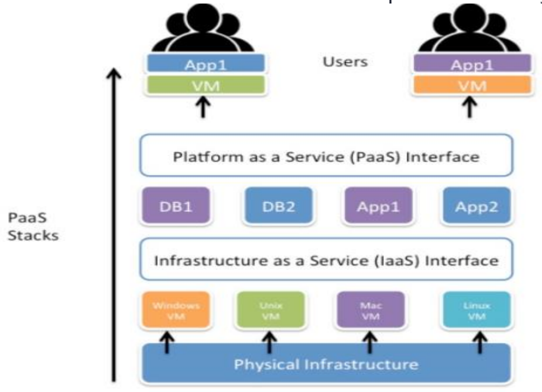
Platform as a service (PaaS) is a cloud-computing and software development paradigm aimed at supporting rapid development. PaaS aims to attach itself to, and extend pre-existing virtualized infrastructure offerings. PaaS adds to these offerings the ability to automatically configure a virtualized environment and install ready-to-use software stacks. By adding the software layer this model provides a highly scalable self-service solution, which can be leveraged for development and delivery purposes. This helps alleviate the long-standing tradition of manually configuring virtual machines, their operating systems and base-level configurations in preparation for development. The primary objective of PaaS or any automation solution is to aid developers by making them more efficient. Abstracting and automating commonly tedious infrastructure and configuration management tasks makes PaaS an alluring development paradigm for modern software organizations.

Effective PaaS implementations provide a set of virtual machines and an A la Carte menu of software stacks, which are configured upon request. Most PaaS service providers will wrap the entire system into a uniquely accessible entity and provide a common interface where end-users can select the desired OS and preferred software stacks. This allows them to spin up environments (complete with attached applications and development infrastructures) via a button click. At the time of its inception the PaaS the paradigm was quite novel and innovative. It expanded on the self-service model of Infrastructure as a Service ("IaaS") and saved significant amounts of time. This in turn made developers more productive.

PaaS is most commonly seen as a logical extension of IaaS. This evolutionary nature is very popular across modern software technologies in engineering. What once started as bare metal hardware slowly evolved into progressive virtualized machines and then finally into micro-sized containers and PaaS offerings. This evolutionary nature has led to many innovations related to virtualized systems, software configuration management and the coupling of the two. The figure below illustrates the historical milestones, which led to the fruition of PaaS based solutions in 2006.


New Hotness
Platform as a Service (PaaS)
Infrastructure as a Service (IaaS)
Virtualized Machine Templates
Manually Spun up Platforms & Environments
Old and Busted
The evolution of Platform as a Service

As the innovative concept of PaaS gains traction and becomes more widely adopted throughout the technology industry, concrete implementation examples and best practices are also evolving. To date there are a number of documentary whitepapers and high-level architectural best practices. The next figure presents a high-level architecture diagram of a PaaS based service. It is based on research of implementations by industry pioneers; including IBM, NetFlix and others.

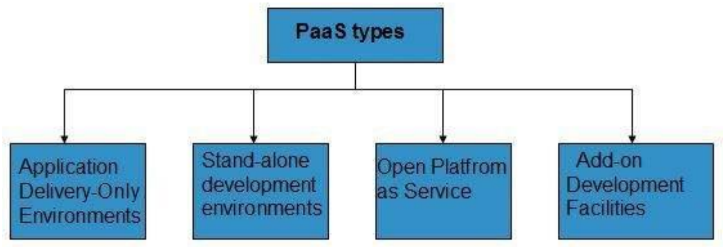


From the diagram above we can visualize PaaS service providers and begin to understand the interconnected interfaces of PaaS, and IaaS. As we can see the base level VMs are provisioned via modern IaaS interfaces and then configuration management software (Ansible, Salt Stack, Chef, Puppet, Vagrant etc.) is leveraged by the PaaS interface to install and manage the requested software solutions. Finally the appropriately requested appliance is delivered to the requesting user via a seamless virtual interface. Once the appliance has served its purpose, it is torn down and removed. This automatic teardown helps the need for IT management of individual VMs and keeps the system scalable.

As PaaS solutions have evolved a new era of container based applications, micro architectures, and PaaS service providers have materialized. The crop up of scalable virtualized solutions and related ancillary tools is a result of the continuing decline in infrastructure costs and the increasing availability of reliable virtualized systems.

PaaS Types

Based on the functions, PaaS can be classified into four types as shown in the following diagram:


PaaS types
Application Delivery-Only Environments
Stand-alone development environments
Open Platfrom as Service
Add-on Development Facilities

Stand-alone development environments
The **stand-alone PaaS** works as an independent entity for a specific function. It does not include licensing or technical dependencies on specific SaaS applications.
Application delivery-only environments
The **application delivery PaaS** includes **on-demand scaling** and **application security.**
Open platform as a service
Open PaaS offers an **open source software** that helps a PaaS provider to run applications.
Add-on development facilities
The **add-on PaaS** allows to customize the existing SaaS platform.

Azure Cloud

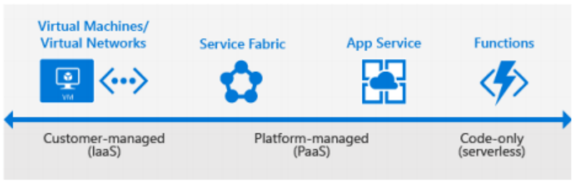
Azure is a complete cloud platform that can host existing applications and streamline new application development. Azure can even enhance on-premises applications. Azure integrates the cloud services that need to develop, test, deploy,and manage applications, all while taking advantage of the efficiencies of cloud computing. By hosting your applications in Azure, start small and easily scale application as customer demand grows. Azure also offers the reliability that's needed for high-availability applications, even including failover between different regions. The Azure portal lets you easily manage all your Azure services. You can also manage services programmatically by using service-specific APIs and templates. This guide is an introduction to the Azure platform for application developers. It provides guidance and direction that need to start building new applications in Azure or migrating existing applications to Azure.

Start Azure

First ,decide on how to host application in Azure. Need to manage entire infrastructure as a virtual machine(VM), use the platform management facilities that Azure provides, need a serverless framework to host code execution only. The application needs cloud storage, which Azure provides several options and can take advantage of Azure's enterprise authentication.

Application hosting

Azure provides several cloud-based compute offerings to run application so that you don't have to worry about the infrastructure details. Applications can easily scale up or scale out resources as application usage grows. Azure offers services that support to application development and hosting needs. Azure provides Infrastructure as a Service(IaaS) to give full control over application hosting. Azure's Platform as a Service(PaaS) offerings provide the fully managed services needed to power apps. There's even true serverless hosting in Azure where all that needed is write code.



Azure App Service

When the quickest path to publish web-based projects, consider Azure App Service. App Service makes it easy to extend web apps to support mobile clients and publish easily consumed REST APIs. This platform provides authentication by using social providers, traffic-based autoscaling, testing in production, and continuous and container-based deployments. We can create web apps, mobileapp back ends,and APIapps. Because all three app types share the App Service runtime ,can host a website, support mobile clients, and expose APIs in Azure, all from the same project or solution.

Azure Virtual Machines

As an Infrastructureas aService(IaaS) provider, Azure lets deploy to or migrate application to either Windows or Linux VMs. Together with Azure Virtual Network, Azure Virtual Machines supports the deployment of Windows or Linux VMs to Azure. With VMs, we have total control over the configuration of the machine. When using VMs, we are responsible for all server software installation, configuration, maintenance, and operating system patches. Because of the level of control that have with VMs, we can run a wide range of server workloads on Azure that don't fit into a PaaS model. These workloads include database servers, Windows Server Active Directory, and Microsoft SharePoint.

Azure Functions (serverless)

Rather than worrying about building out and managing a whole application or the infrastructure to run code, what if we could just write the code and have it run in response to events or on a schedule? Azure Functions is a "serverless"-style offering that lets you write just the code needed. With Functions, we can trigger code execution with HTTP requests, web hooks, cloud service events, or on a schedule. We can code in development language of choice, such as C#,F#, Node.js, Python, or PHP. With consumption-based billing, you pay only for the time that code executes, and Azure scales as needed.

Azure Service Fabric

Azure Service Fabric is a distributed systems platform. This platform makes it easy to build, package, deploy, and manage scalable and reliable microservices. It also provides comprehensive application management capabilities such as: Provisioning Deploying Monitoring Upgrading/patching Deleting Apps, which run on a shared pool of machines, can start small and scaleto hundreds or thousands of machines as needed.

Azure Spring Cloud

Azure Spring Cloud is a serverless microservices platform that enables to build, deploy, scale and monitor your applications in the cloud. Use Spring Cloud to bring modern microservice patterns to Spring Bootapps, eliminating boiler plate code to quickly build robust Java apps.

- Leverage managed versions of Spring Cloud Service Discovery and Config Server, while we ensure those critical components are running in optimum conditions.
- Focus on building your business logic and we will take care of your service runtime with security patches, compliance standards and high availability.
- Manage application lifecycle(e.g.: deploy, start, stop, scale) on top of Azure Kubernetes Service.
- Easily bind connections between your apps and Azure services such as Azure Database for MySQL and Azure Cache for Redis.
- Monitor and troubleshoot micro services and applications using enterprise-grade unified monitoring tools that offer deep insights on application dependencies and operational telemetry.
- Azure Storage: Offers durable, highly available storagefor blobs, queues, files, and other kinds of non relational data. Storage provides the storage foundation for VMs
- Azure SQL Database: An Azure-based version of the MicrosoftSQL Server engine for storing relational tabular data in thecloud.SQL Database provides predictable performance, scalability with no downtime, business continuity, and data protection.

Aneka Cloud

Aneka is a software platform and a framework for developing distributed applications on the cloud. It harnesses the computing resources of a heterogeneous network of workstations and servers or data centers on demand. Aneka provides developers with a rich set of APIs for transparently exploiting these resources by expressing the application logic with a variety of programming abstractions. System administrators can leverage a collection of tools to monitor and control the deployed infrastructure. This can be a public cloud available to anyone through the Internet, a private cloud constituted by a set of nodes with restricted access within an enterprise, or a hybrid cloud where external resources are integrated on demand, thus allowing applications to scale. Figure 9.1 provides a layered view of the framework. Aneka is essentially an implementation of the PaaS model, and it provides a runtime environment for executing applications by leveraging the underlying infrastructure of the cloud. Developers can express distributed applications by using the API contained in the Software Development Kit (SDK) or by porting existing legacy applications to the cloud. Such applications are executed on the Aneka cloud, represented by a collection of nodes connected through the network hosting the Aneka container. The container is the building block of the middleware and represents the runtime environment for executing applications; it contains the core functionalities of the system and is built up from an extensible collection of services that allow administrators to customize the Aneka cloud. There are three classes of services that characterize the container:

- Execution Services. They are responsible for scheduling and executing applications. Each of the programming models supported by Aneka defines specialized implementations of these services for managing the execution of a unit of work defined in the model.
- Foundation Services. These are the core management services of the Aneka container. They are in charge of metering applications, allocating resources for execution, managing the collection of available nodes, and keeping the services registry updated.
- Fabric Services: They constitute the lowest level of the services stack of Aneka and provide access to the resources managed by the cloud. An important service in this layer is the Resource Provisioning Service, which enables horizontal scaling3 in the cloud. Resource provisioning makes Aneka elastic and allows it to grow or to shrink dynamically to meet the QoS requirements of applications.

  The container relies on a platform abstraction layer that interfaces it with the underlying host, whether this is a physical or a virtualized resource. Aneka also provides a tool for managing the cloud, allowing administrators to easily start, stop, and deploy instances of the Aneka container on new resources and then reconfigure them dynamically to alter the behavior of the cloud.

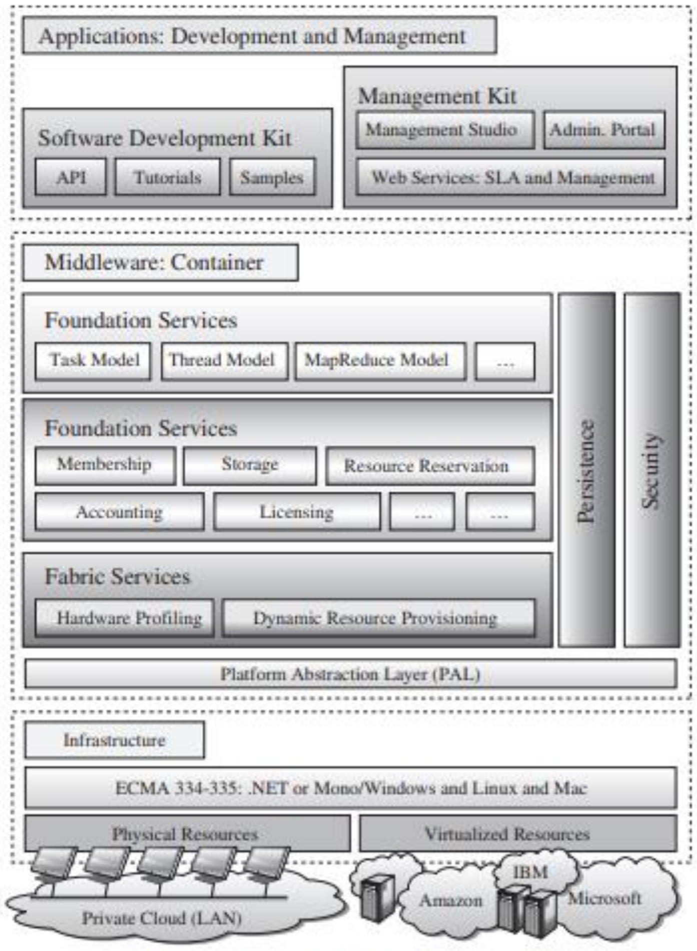


FIGURE 9.1. Aneka framework architecture.

ANEKA RESOURCE PROVISIONING SERVICE

The most significant benefit of cloud computing is the elasticity of resources, services, and applications, which is the ability to automatically scale out based on demand and users' quality of service requests. Aneka as a PaaS not only features multiple programming models allowing developers to easily build their distributed applications, but also provides resource provisioning facilities in a seamless and dynamic fashion. Applications managed by the Aneka container can be dynamically mapped to heterogeneous resources, which can grow or shrink according to the application's needs. This elasticity is achieved by means of the resource provisioning framework, which is composed primarily of services built into the Aneka fabric layer. Figure 9.2 provides an overview of Aneka resource provisioning over private and public clouds. This is a typical scenario that a medium or large enterprise may encounter; it combines privately owned resources with public rented resources to dynamically increase the resource capacity to a larger scale. Private resources identify computing and storage elements kept in the premises that share similar internal security and administrative policies. Aneka identifies two types of private resources: static and dynamic resources. Static resources are constituted by existing physical workstations and servers that may be idle for a certain period of time. Their membership to the Aneka cloud is manually configured by administrators and does not change over time. Dynamic resources are mostly represented by virtual instances that join and leave the Aneka cloud and are controlled by resource pool managers that provision and release them when needed. Horizontal scaling is the process of adding more computing nodes to a system. It is counter posed to vertical scaling, which is the process of increasing the computing capability of a single computer resource.

Public resources reside outside the boundaries of the enterprise and are provisioned by establishing a service-level agreement with the external provider. Even in this case we can identify two classes: on-demand and reserved resources. On-demand resources are dynamically provisioned by resource pools for a fixed amount of time (for example, an hour) with no long-term commitments and on a pay-as-you-go basis. Reserved resources are provisioned in advance by paying a low, one-time fee and mostly suited for long-term usage. These resources are actually the same as static resources, and no automation is needed in the resource provisioning service to manage them. Despite the specific classification previously introduced, resources are managed uniformly once they have joined the Aneka cloud and all the standard operations that are performed on statically configured nodes can be transparently applied to dynamic virtual instances. Moreover, specific operations pertaining to dynamic resources, such as join and leave, are seen as connection and disconnection of nodes and transparently handled. This is mostly due to the indirection layer provided by the Aneka container that abstracts the specific nature of the hosting machine.

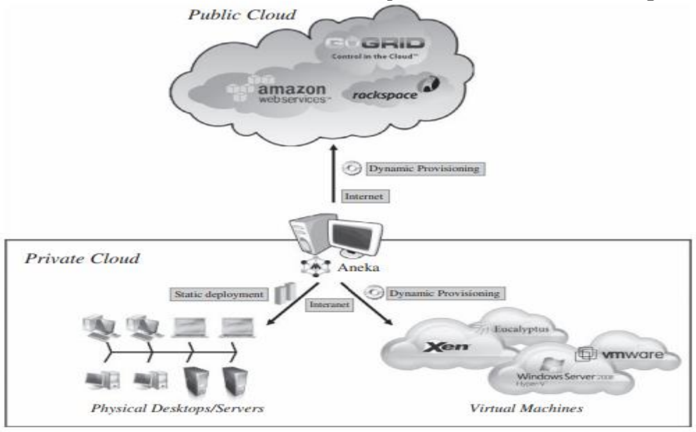


FIGURE 9.2. Aneka resource provisioning over private and public clouds.

Aneka Hybrid Cloud Architecture

The Resource Provisioning Framework represents the foundation on top of which Aneka-based hybrid clouds are implemented. In this section we will introduce the components that compose this framework and briefly describe their interactions. The basic idea behind the Resource Provisioning Framework is depicted in Figure 9.4. The resource provisioning infrastructure is represented by a collection of resource pools that provide access to resource providers, whether they are external or internal, and managed uniformly through a specific component called a resource pool manager. A detailed description of the components follows:

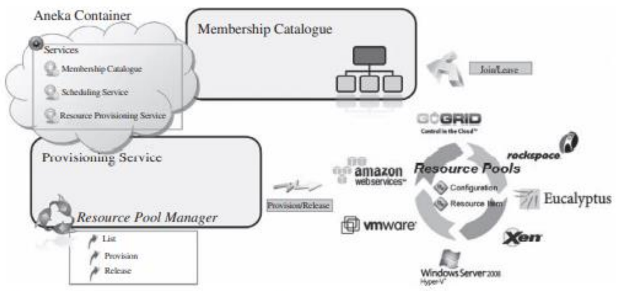


FIGURE 9.4. System architecture of the Aneka Resource Provisioning Framework.

- Resource Provisioning Service. This is an Aneka-specific service that implements the service interface and wraps the resource pool manager, thus allowing its integration within the Aneka container.
- Resource Pool Manager. This manages all the registered resource pools and decides how to allocate resources from those pools. The resource pool manager provides a uniform interface for requesting additional resources from any private or public provider and hides the complexity of managing multiple pools to the Resource Provisioning Service.
- Resource Pool. This is a container of virtual resources that mostly come from the same resource provider. A resource pool is in charge of managing the virtual resources it contains and eventually releasing them when they are no longer in use. Since each vendor exposes its own specific interfaces, the resource pool (a) encapsulates the specific implementation of the communication protocol required to interact with it and (b) provides the pool manager with a unified interface for acquiring, terminating, and monitoring virtual resources

The request for additional resources is generally triggered by a scheduler that detects that the current capacity is not sufficient to satisfy the expected quality of services ensured for specific applications. In this case a provisioning request is made to the Resource Provisioning Service. According to specific policies, the pool manager determines the pool instance(s) that will be used to provision resources and will forward the request to the selected pools. Each resource pool will translate the forwarded request by using the specific protocols required by the external provider and provision the resources. Once the requests are successfully processed, the requested number of virtual resources will join the Aneka cloud by registering themselves with the Membership Catalogue Service, which keeps track of all the nodes currently connected to the cloud. Once joined the cloud the provisioned resources are managed like any other node. A release request is triggered by the scheduling service when provisioned resources are no longer in use. Such a request is then forwarded to the interested resources pool (with a process similar to the one described in the previous paragraph) that will take care of terminating the resources when more appropriate. A general guideline for pool implementation is to keep provisioned resources active in a local pool until their lease time expires. By doing this, if a new request arrives within this interval, it can be served without leasing additional resources from the public infrastructure. Once a virtual instance is terminated, the Membership Catalogue Service will detect a disconnection of the corresponding node and update its registry accordingly. It can be noticed that the interaction flow previously described is completely independent from the specific resource provider that will be integrated into the system. In order to satisfy such a requirement, modularity and welldesigned interfaces between components are very important. The current design, implemented in Aneka, maintains the specific implementation details within the Resource Pool implementation, and resource pools can be dynamically configured and added by using the dependency injection techniques, which are already implemented for configuring the services hosted in the container.