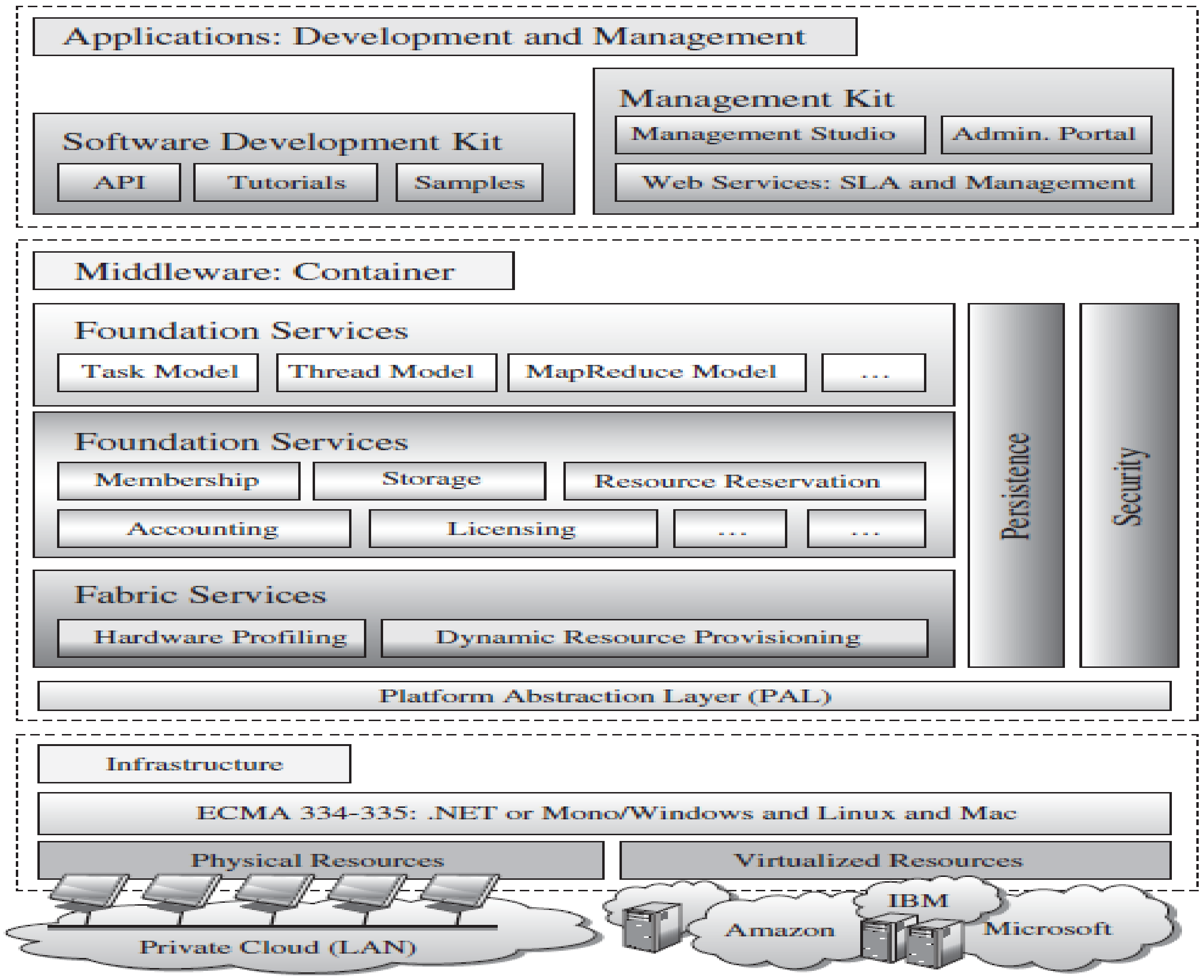
ANEKA—INTEGRATION OF PRIVATE AND PUBLIC CLOUDS

ANEKA CLOUD PLATFORM

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 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:**

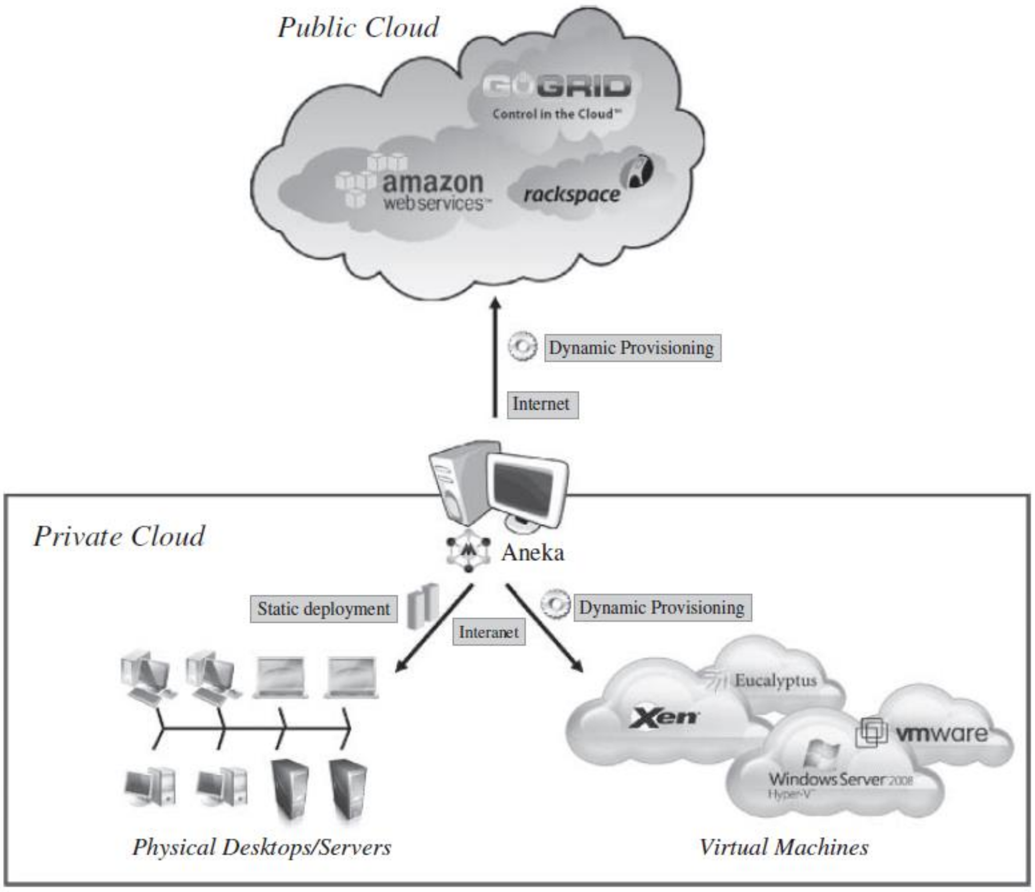
They constitute the lowest level of the services stack of Aneka and provide access to the resources managed by the cloud. An important service in this layer is the Resource Provisioning Service, which enables horizontal scaling3 in the cloud.
Resource provisioning makes Aneka elastic and allows it to 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. This makes the container portable over different runtime environments that feature an implementation of the ECMA 334 [23] and ECMA 335 [24] specifications (such as the .NET framework or Mono).

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.

ANEKA RESOURCE PROVISIONING SERVICE

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:

1. **Static resources and**
2. **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.

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:**

1. **On-demand and**
2. **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.

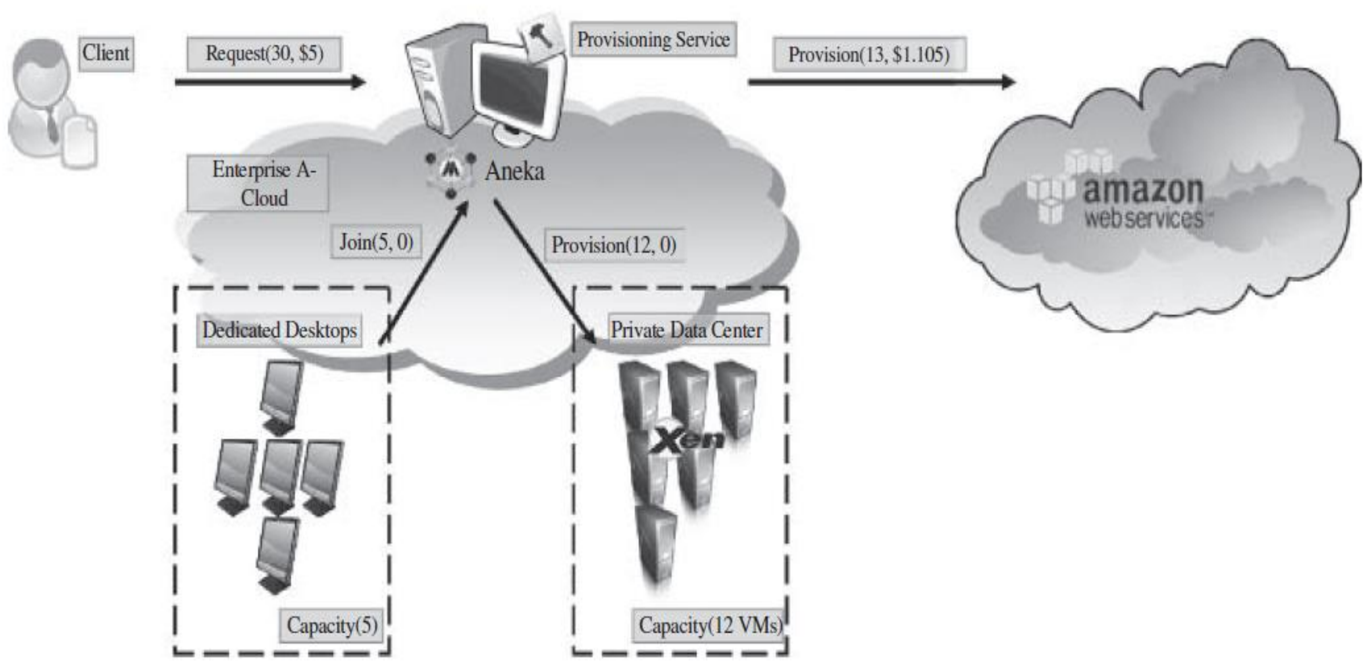
Reserved resources are provisioned in advance by paying a low, one-time fee and mostly suited for long-term usage. These resources are actually the same as static resources, and no automation is needed in the resource provisioning service to manage them.

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.

Resource Provisioning Scenario

Figure 9.3 illustrates a possible scenario in which the resource provisioning service becomes important.



A private enterprise maintains a private cloud, which consists of:
(a) Five physical dedicated desktops from its engineering department and
(b) a small data center managed by Xen Hypervisor providing virtual machines with the maximum capacity of 12 VMs.

In most of the cases, this setting is able to address the computing needs of the enterprise. **In the case of peak computing demand**, additional resources can be provisioned by leveraging the virtual public infrastructure.

For example, a mission critical application could require at least 30 resources to complete within an hour, and the customer is willing to spend a maximum of 5 dollars to achieve this goal.

In this case, the Aneka Resource Provisioning service becomes a fundamental infrastructure component to address this scenario.

In this case, once the client has submitted the application, the Aneka scheduling engine detects that the current capacity in terms of resources (5 dedicated nodes) is not enough to satisfy the user's QoS requirement and to complete the application on time.

An additional 25 resources must be provisioned. It is the responsibility of the Aneka Resource Provisioning service to acquire these resources from both the private data center managed by Xen Hypervisor and the Amazon public cloud.

The provisioning service is configured by default with a cost-effective strategy, which privileges the use of local resources instead of the dynamically provisioned and chargeable ones.
The computing needs of the application require the full utilization of the local data center that provides the Aneka cloud with 12 virtual machines. Such capacity is still not enough to complete the mission critical application in time; and the remaining 13 resources are rented from Amazon for a minimum of one hour, which incurs a few dollars' cost.

This is not the only scenario that Aneka can support, and different provisioning patterns can be implemented.

Another simple strategy for provisioning resources could be minimizing the execution time to let the application finish as early as possible; this requires Aneka to request more powerful resources from the Amazon public cloud.

For example, in the previous case instead of provisioning 13 small instances from Amazon, a major number of resources, or more powerful resources, can be rented by spending the entire budget available for the application.

The resource provisioning infrastructure can also serve broader purposes such as keeping the length of the system queue, or the average waiting time of a job in the queue, under a specified value. In these cases, specific policies can be implemented to ensure that the throughput of the system is kept at a reasonable level.

HYBRID CLOUD IMPLEMENTATION

Currently, there is no widely accepted standard for provisioning virtual infrastructure from Infrastructure as a Service (IaaS) providers, but each provider exposes its own interfaces and protocols. Hence, it is not possible to seamlessly integrate different providers into one single infrastructure.

The resource provisioning service implemented in Aneka addresses these issues and abstracts away the differences of providers' implementation.

Design and Implementation Guidelines

The particular nature of hybrid clouds demands additional and specific functionalities that software engineers have to consider while designing software systems supporting the execution of applications in hybrid and dynamic environments. **These features, together with some guidelines on how to implement them, are presented in the following:**

_ **Support for Heterogeneity**. Hybrid clouds are produced by heterogeneous resources such as clusters, public or private virtual infrastructures, and workstations. In particular, for what concerns a virtual machine manager, it must be possible to integrate additional cloud service providers (mostly IaaS providers) without major changes to the entire system design and codebase. Hence, the specific code related to a particular cloud resource provider should be kept isolated behind interfaces and within pluggable components.

_ **Support for Dynamic and Open Systems**. Hybrid clouds change their composition and topology over time. They form as a result of dynamic conditions such as peak demands or specific Service Level Agreements attached to the applications currently in execution. An open and extensible architecture that allows easily plugging new components and rapidly integrating new features is of a great value in this case. Specific enterprise architectural patterns can be considered while designing such software systems. In particular, inversion of control and, more precisely, dependency injection5 in component-based systems is really helpful.

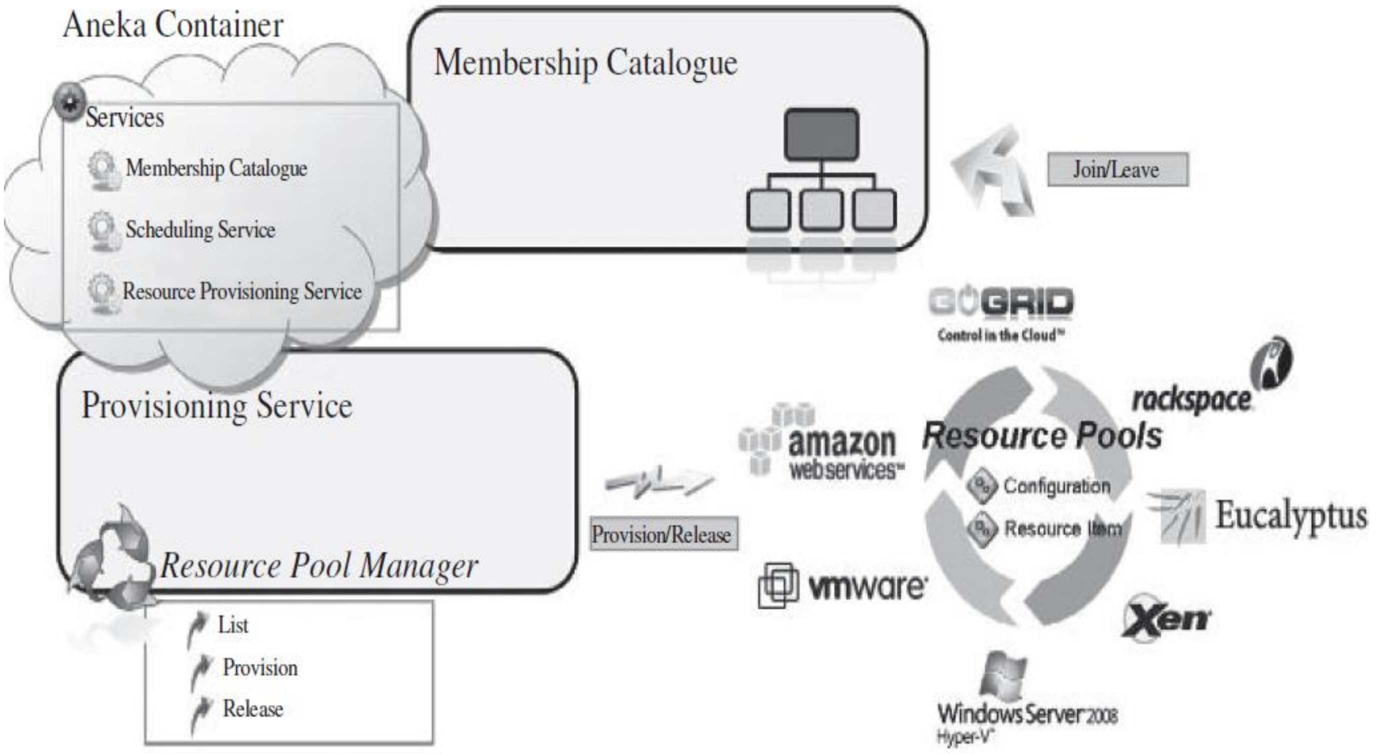
_ **Support for Basic VM Operation Management**. Hybrid clouds integrate virtual infrastructures with existing physical systems. Virtual infrastructures are produced by virtual instances. Hence, software frameworks that support hypervisor-based execution should implement a minimum set of operations. They include requesting a virtual instance, controlling its status, terminating its execution, and keeping track of all the instances that have been requested.

_ **Support for Flexible Scheduling Policies**. The heterogeneity of resources that constitute a hybrid infrastructure naturally demands for flexible scheduling policies. Public and private resources can be differently utilized, and the workload should be dynamically partitioned into different streams according to their security and quality of service (QoS) requirements. There is then the need of being able to transparently change scheduling policies over time with a minimum impact on the existing infrastructure and almost now downtimes. Configurable scheduling policies are then an important feature.

_ **Support for Workload Monitoring**. Workload monitoring becomes even more important in the case of hybrid clouds where a subset of resources is leased and resources can be dismissed if they are no longer necessary. Workload monitoring is an important feature for any distributed middleware, in the case of hybrid clouds, it is necessary to integrate this feature with scheduling policies that either directly or indirectly govern the management of virtual instances and their leases.

Aneka Hybrid Cloud Architecture

The Resource Provisioning Framework represents the foundation on top of which Aneka-based hybrid clouds are implemented. **The basic idea behind the Resource Provisioning Framework is depicted in Figure**



System architecture of the Aneka Resource Provisioning Framework

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:**

- **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.

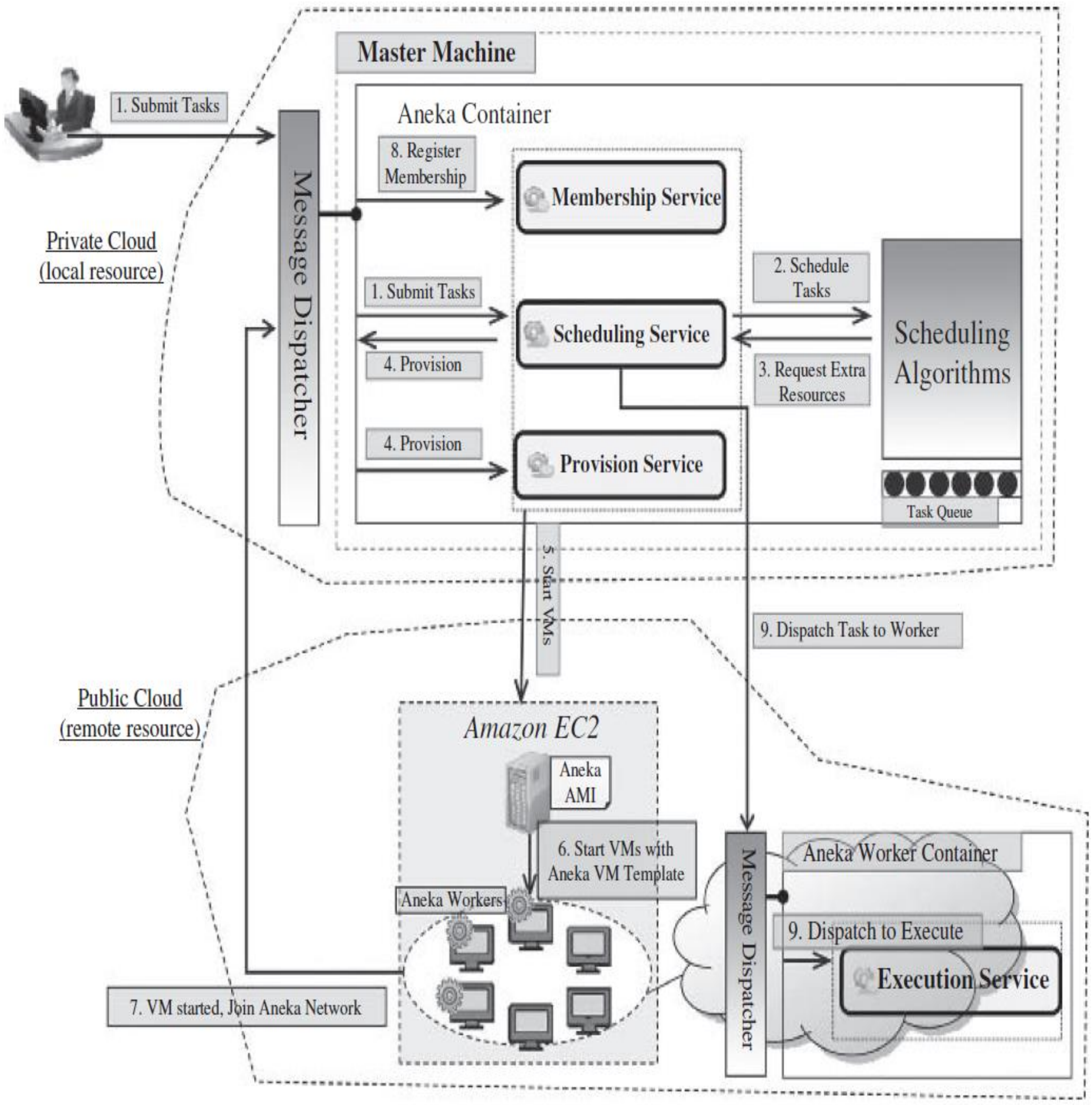
The current implementation of Aneka allows customizing the Resource Provisioning Infrastructure by specifying the following elements:

- **Resource Provisioning Service**. The default implementation provides a lightweight component that generally forwards the requests to the resource Pool Manager. A possible extension of the system can be the implementation of a distributed resource provisioning service that can operate at this level or at the Resource Pool Manager level.
- **Resource Pool Manager**. The default implementation provides the basic management features required for resource and provisioning request forwarding.
- **Resource Pools.** The Resource Pool Manager exposes a collection of resource pools that can be used. It is possible to add any implementation that is compliant to the interface contract exposed by the Aneka provisioning API, thus adding a heterogeneous open-ended set of external providers to the cloud.

- **Provisioning Policy**. Scheduling services can be customized with resource provisioning aware algorithms that can perform scheduling of applications by taking into account the required QoS.

Implementation Steps for Aneka Resource Provisioning Service

The resource provisioning service is a customized service which will be used to enable cloud bursting by Aneka at runtime.
Figure demonstrates one of the application scenarios that utilize resource provisioning to dynamically provision virtual machines from Amazon EC2 cloud.



Aneka resource provisioning (cloud bursting) over Amazon EC2

The general steps of resource provisioning on demand in Aneka are the following:

- The application submits its tasks to the scheduling service, which, in turns, adds the tasks into the scheduling queue.
- The scheduling algorithm finds an appropriate match between a task and a resource. If the algorithm could not find enough resources for serving all the tasks, it requests extra resources from the scheduling service.
- The scheduling service will send a ResourceProvisionMessage to provision service and will ask provision service to get X number of resources as determined by the scheduling algorithm.

- Upon receiving the provision message, the provision service will delegate the provision request to a component called resource pool manager, which is responsible for managing various resource pools.
  A resource pool is a logical view of a cloud resource provider, where the virtual machines can be provisioned at runtime. Aneka resource provisioning supports multiple resource pools such as Amazon EC2 pool and Citrix Xen server pool.
- The resource pool manager knows how to communicate with each pool and will provision the requested resources on demand. Based on the requests from the provision service, the pool manager starts X virtual machines by utilizing the predefined virtual machine template already configured to run Aneka containers.
- A worker instance of Aneka will be configured and running once a virtual resource is started. All the work instances will then connect to the Aneka master machine and will register themselves with Aneka membership service.
- The scheduling algorithm will be notified by the membership service once those work instances join the network, and it will start allocating pending tasks to them immediately.
- Once the application is completed, all the provisioned resources will be released by the provision service to reduce the cost of renting the virtual machine.