UNIT VI-Future of Cloud Computing

How the Cloud Will Change Operating Systems, Location-Aware Applications, Intelligent Fabrics, Paints, and More, The Future of Cloud TV, Future of Cloud-Based Smart Devices, Faster Time to Market for Software Applications, Home-Based Cloud Computing, Mobile Cloud, Autonomic Cloud Engine, Multimedia Cloud, Energy Aware Cloud Computing, Jungle Computing.

Docker at a Glance: Process Simplification, Broad Support and Adoption, Architecture, Getting the Most from Docker, The Docker Workflow

Mobile Cloud Computing (MCC)

- *Mobile cloud computing (MCC)* at its simplest, refers to an infrastructure where both the data storage and data processing happen outside of the mobile device.

- Mobile cloud applications move the computing power and data storage away from the mobile devices and into powerful and centralized computing platforms located in clouds, which are then accessed over the wireless connection based on a thin native client.

Mobile Cloud Computing (MCC)

- Mobile devices face many resource challenges (battery life, storage, bandwidth etc.)

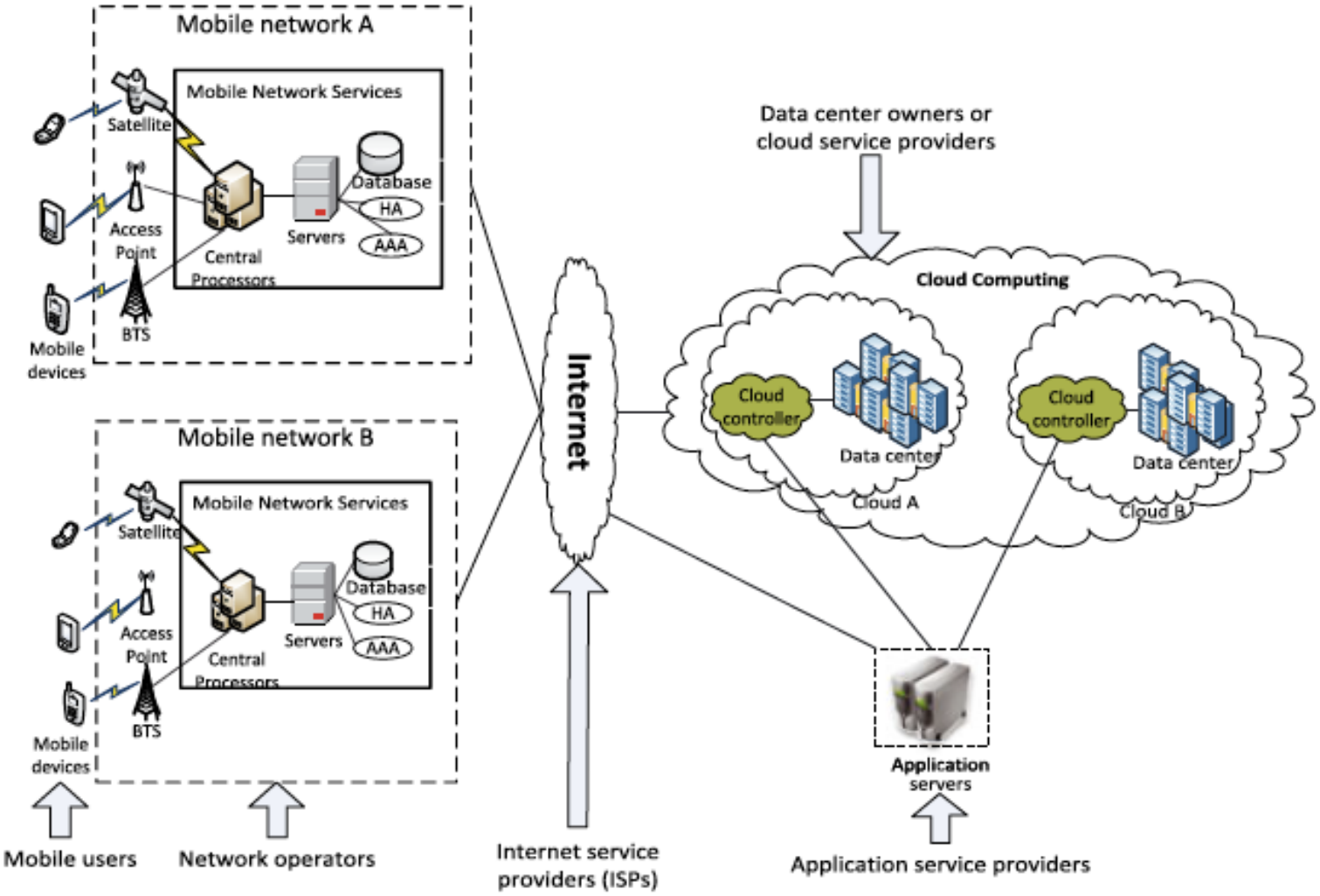
- Cloud computing offers advantages to users by allowing them to use infrastructure, platforms and software by cloud providers at low cost and elastically in an on-demand fashion.

- Mobile cloud computing provides mobile users with data storage and processing services in clouds, obviating the need to have a powerful device configuration (e.g. CPU speed, memory capacity etc), as all resource-intensive computing can be performed in the cloud.

Mobile Cloud Computing (MCC)

- According to a recent study by ABI Research, more than 240 million business will use cloud services through mobile devices by 2015.
- That traction will push the revenue of mobile cloud computing to $5.2 billion.
- Mobile cloud computing is a highly promising trend for the future of mobile computing.

MCC Architecture



Mobile Cloud Computing (MCC)

- Mobile devices are connected to the mobile networks via base stations that establish and control the connections and functional interfaces between the networks and mobile devices.

- Mobile users' requests and information are transmitted to the central processors that are connected to servers providing mobile network services.

- The subscribers' requests are delivered to a cloud through the Internet. In the cloud, cloud controllers process the requests to provide mobile users with the corresponding cloud services.

Advantages of MCC

- Extending battery lifetime:
  - Computation offloading migrates large computations and complex processing from resource-limited devices (i.e., mobile devices) to resourceful machines (i.e., servers in clouds).
  - Remote application execution can save energy significantly.
  - Many mobile applications take advantages from task migration and remote processing.

Advantages of MCC

- Improving data storage capacity and processing power:
  - MCC enables mobile users to store/access large data on the cloud.
  - MCC helps reduce the running cost for computation intensive applications.
  - Mobile applications are not constrained by storage capacity on the devices because their data now is stored on the cloud.

Advantages of MCC

- Improving reliability and availability:
  - Keeping data and application in the clouds reduces the chance of lost on the mobile devices.
  - MCC can be designed as a comprehensive data security model for both service providers and users:
    - Protect copyrighted digital contents in clouds.
    - Provide security services such as virus scanning, malicious code detection, authentication for mobile users.
  - With data and services in the clouds, then are always(almost) available even when the users are moving.

Advantages of MCC

- Dynamic provisioning:
  - Dynamic on-demand provisioning of resources on a fine-grained, self-service basis
  - No need for advanced reservation

- Scalability:
  - Mobile applications can be performed and scaled to meet the unpredictable user demands
  - Service providers can easily add and expand a service

Advantages of MCC

- Multi-tenancy:
  - Service providers can share the resources and costs to support a variety of applications and large no. of users.

- Ease of Integration:
  - Multiple services from different providers can be integrated easily through the cloud and the Internet to meet the users' demands.

MCC Applications

- Mobile Commerce:
  - M-commerce allows business models for commerce using mobile devices.
  - Examples: Mobile financial, mobile advertising, mobile shopping…
  - M-commerce applications face various challenges (low bandwidth, high complexity of devices, security, …)
  - Integrated with cloud can help address these issues
  - Example: Combining 3G and cloud to increase data processing speed and security level.

MCC Applications

- Mobile Learning:
  - M-learning combines e-learning and mobility
  - Traditional m-learning has limitations on high cost of devices/network, low transmission rate, limited educational resources
  - Cloud-based m-learning can solve these limitations
  - Enhanced communication quality between students and teachers
  - Help learners access remote learning resources
  - A natural environment for collaborative learning

MCC Applications

- Mobile Healthcare:
  - M-healthcare is to minimize the limitations of traditional medical treatment (eg. Small storage, security/privacy, medical errors, …)
  - M-healthcare provides mobile users with convenient access to resources(eg. medical records)
  - M-healthcare offers hospitals and healthcare organizations a variety of on-demand services on clouds
  - Examples:
    - Comprehensive health monitoring services
    - Intelligent emergency management system
    - Health-aware mobile devices (detect pulse-rate, blood pressure, level of alcohol etc)
    - Pervasive access to healthcare information
    - Pervasive lifestyle incentive management (to manage healthcare expenses)

MCC Applications

- Mobile Gaming:
  - M-game is a high potential market generating revenues for service providers.
  - Can completely offload game engine requiring large computing resource (e.g., graphic rendering) to the server in the cloud.
  - Offloading can also save energy and increase game playing time (eg. MAUI allows fine-grained energy-aware offloading of mobile codes to a cloud)
  - Rendering adaptation technique can dynamically adjust the game rendering parameters based on communication constraints and gamers' demands

MCC Applications

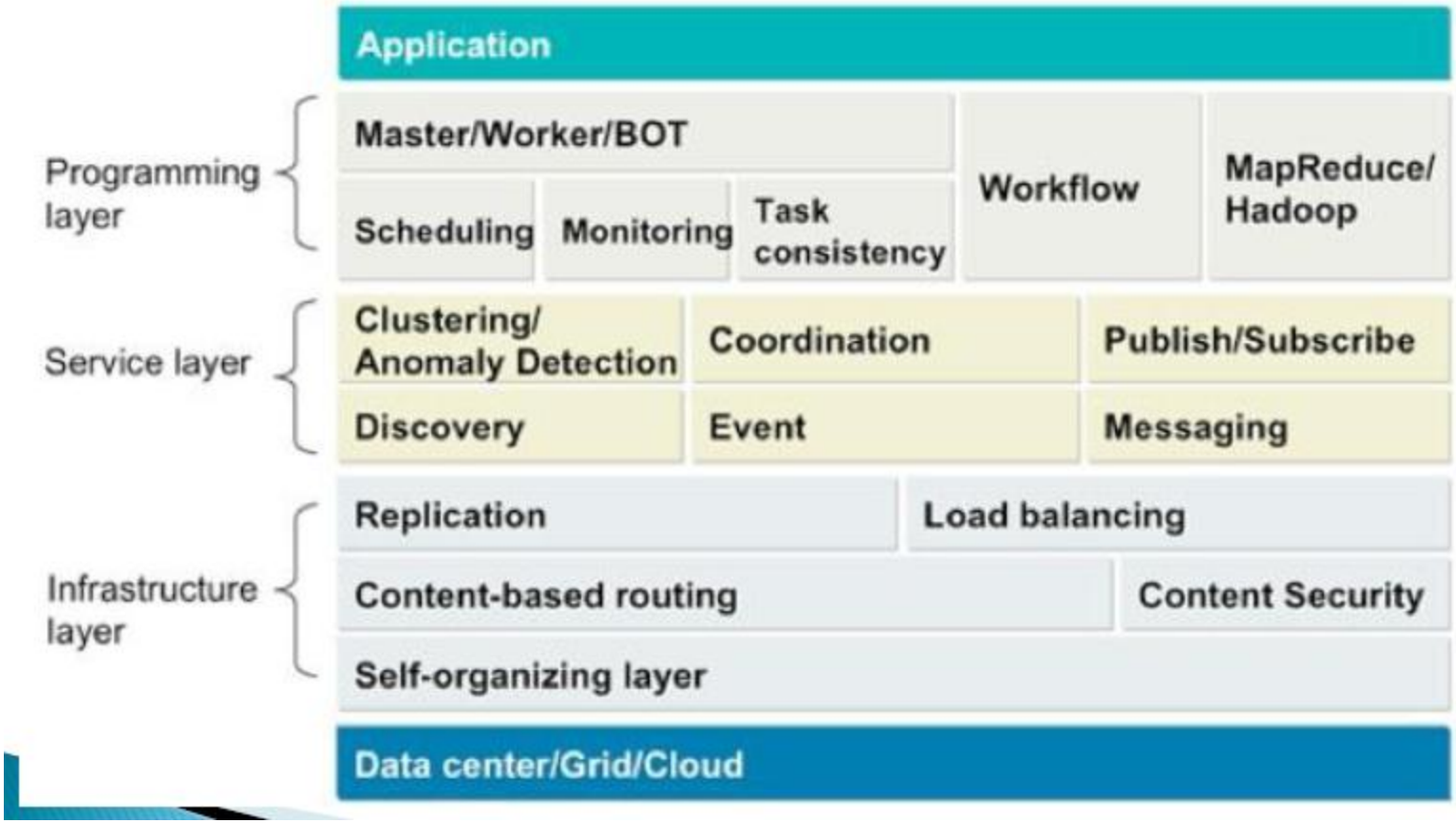
- Assistive technologies:
  - Pedestrian crossing guide for blind and visually-impaired
  - Mobile currency reader for blind and visually impaired
  - Lecture transcription for hearing impaired students

- Other applications:
  - Sharing photos/videos
  - Keyword-based, voice-based, tag-based searching
  - Monitoring a house, smart home systems
  - …

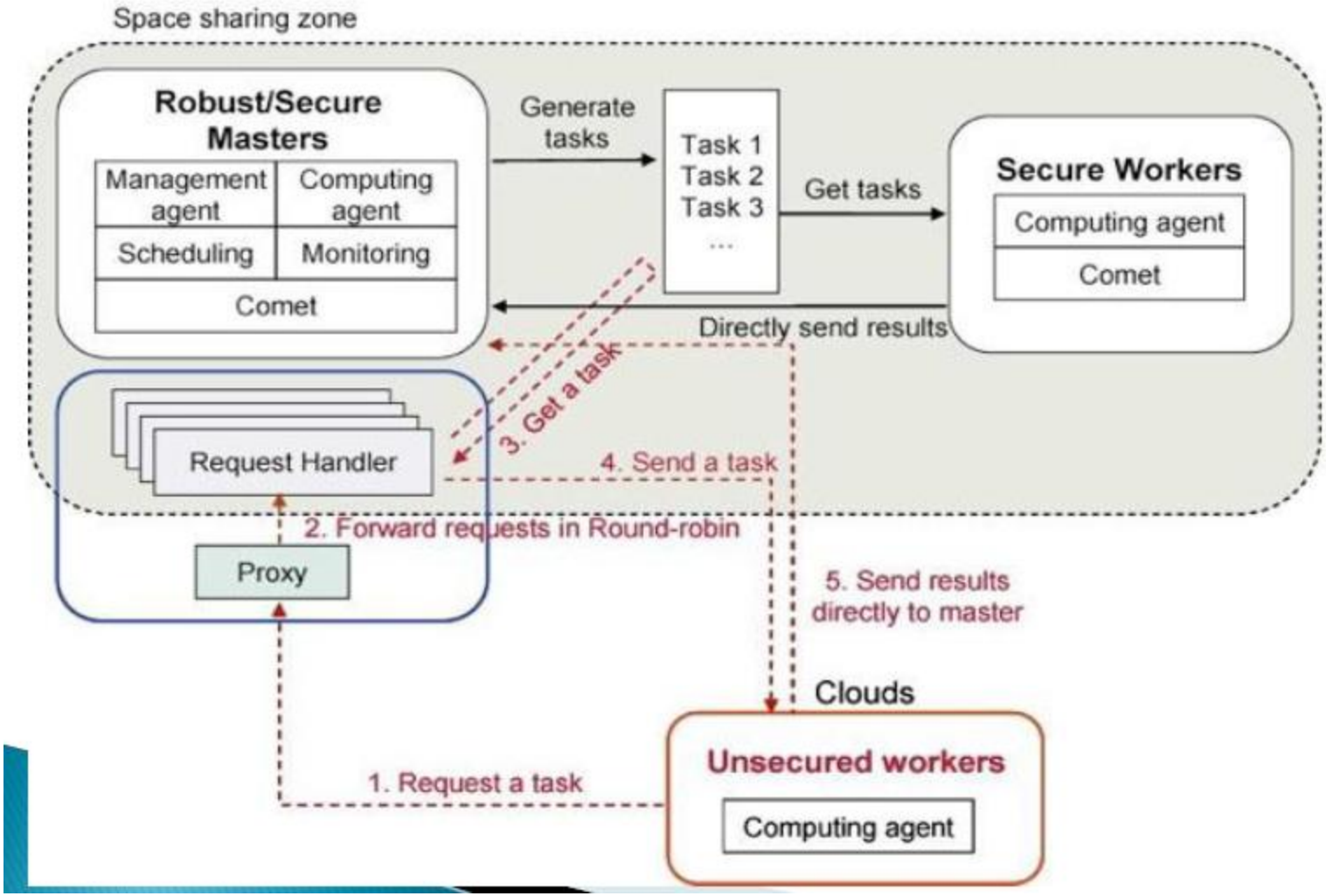
CometCloud

- Integrates of public and private cloud
- Is a PaaS
- to enable on-demand scale-up, scale-down and scale-out
- Cloudbursting
- Cloudbridging

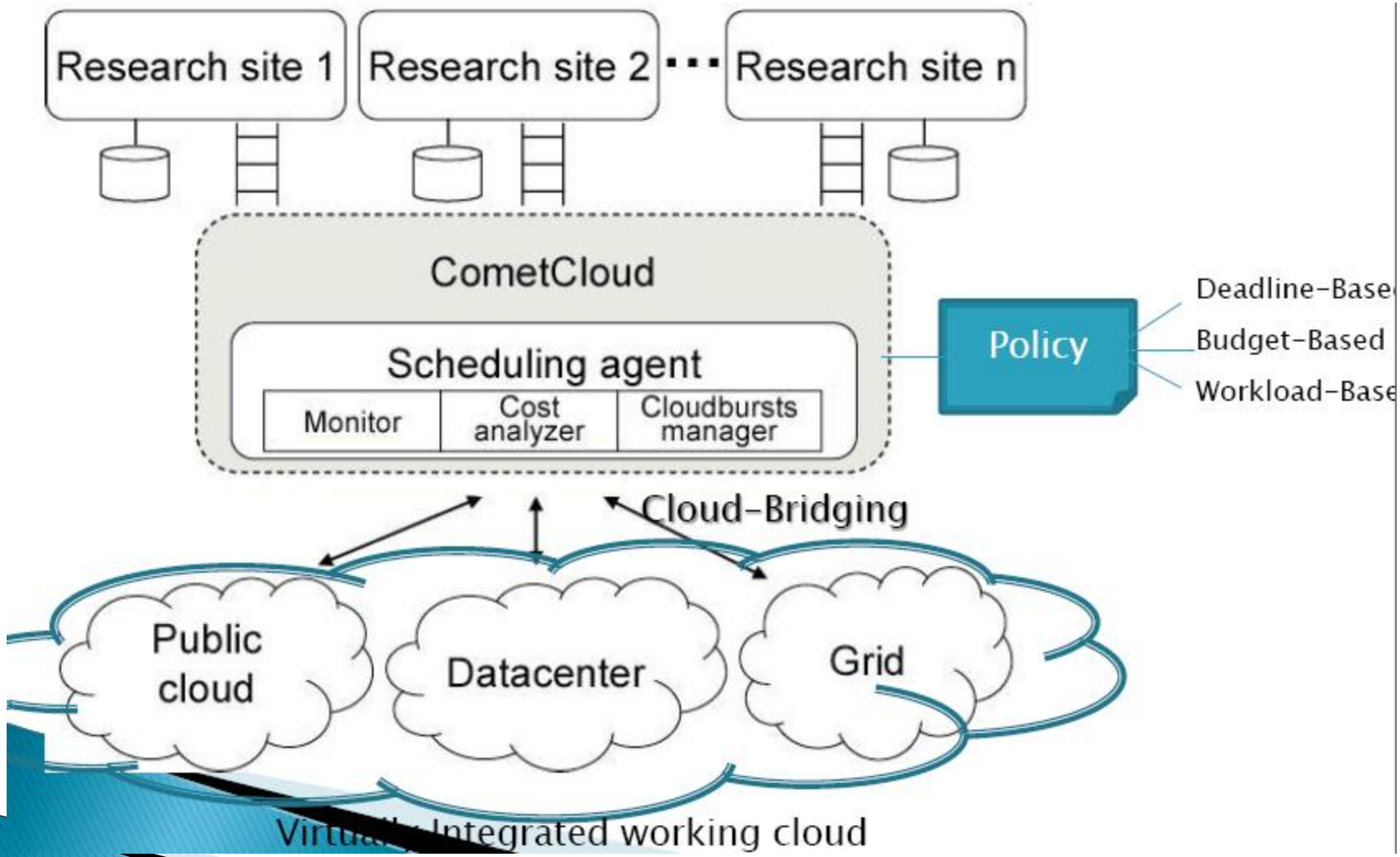
CometCloud


Application
Programming layer
Master/Worker/BOT
Scheduling
Monitoring
Task consistency
Workflow
MapReduce/ Hadoop
Service layer
Clustering/ Anomaly Detection
Coordination
Publish/Subscribe
Discovery
Event
Messaging
Infrastructure layer
Replication
Load balancing
Content-based routing
Content Security
Self-organizing layer
Data center/Grid/Cloud

Automatic Cloud bursting



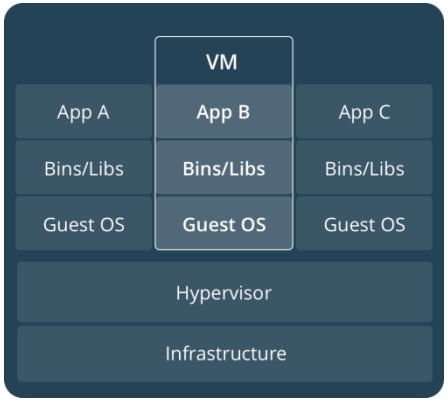
Automatic Cloudbridging

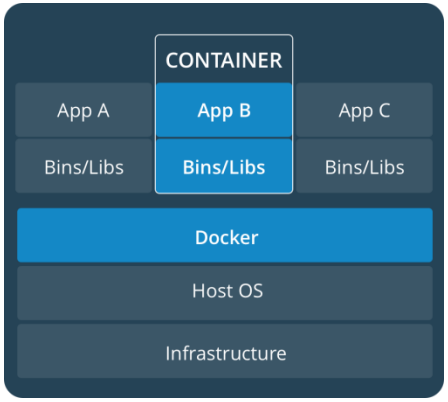


Docker & Container

- Docker is a software platform that allows you to build, test, and deploy applications quickly, packaging software into standardized units called **containers**.

- What is container :
  - Container ≠ VM
  - Isolated
  - Share OS
  - and sometimes bins/libs





Docker

- Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.

- With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

Docker

- Docker provides tooling and a platform to manage the lifecycle of your containers:
  - Develop your application and its supporting components using containers.
  - The container becomes the unit for distributing and testing your application.
  - When you're ready, deploy your application into your production environment, as a container or an orchestrated service. This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two.

Advantages of Docker

- **Fast, consistent delivery of your applications**
- **Responsive deployment and scaling**
- **Running more workloads on the same hardware**

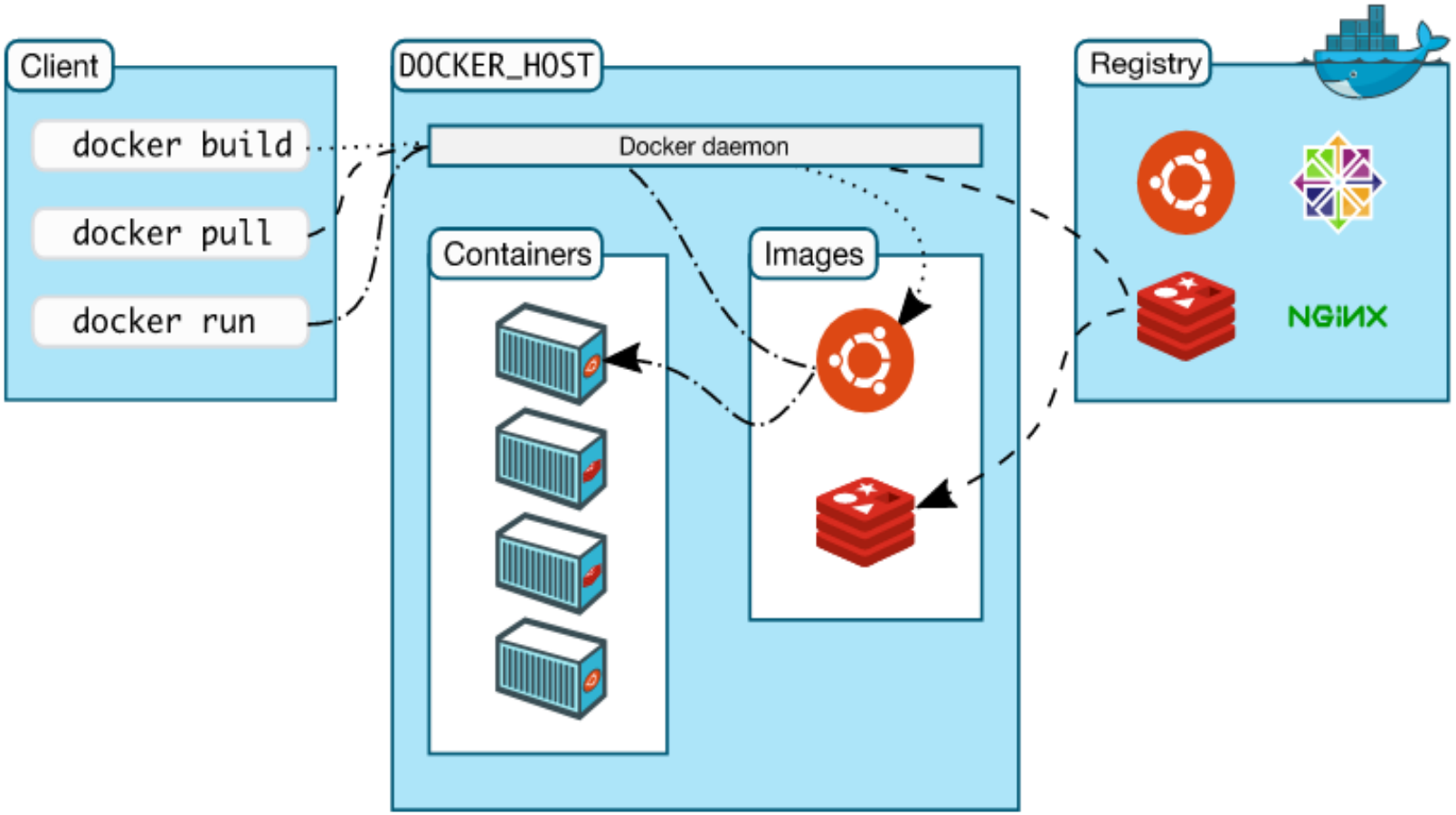
Docker Architecture

- Docker uses a client-server architecture. The Docker *client* talks to the Docker *daemon*, which does the heavy lifting of building, running, and distributing your Docker containers.

- The Docker client and daemon *can* run on the same system, or you can connect a Docker client to a remote Docker daemon.

- The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface. Another Docker client is Docker Compose, that lets you work with applications consisting of a set of containers.

Docker Architecture



Docker Architecture

- The Docker daemon
- The Docker client
- Docker Desktop
- Docker registries
- Docker objects

Docker Architecture

- **The Docker daemon**
  - The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

- **The Docker client**
  - The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out.
    The docker command uses the Docker API. The Docker client can communicate with more than one daemon.

Docker Architecture

- **Docker Desktop**
  - Docker Desktop is an easy-to-install application for your Mac or Windows environment that enables you to build and share containerized applications and microservices.
  - Docker Desktop includes the Docker daemon (dockerd), the Docker client (docker), Docker Compose, Docker Content Trust, Kubernetes, and Credential Helper. For more information, see Docker Desktop.

Docker Architecture

- **Docker registries**
  - A Docker *registry* stores Docker images. Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.
  - When you use the docker pull or docker run commands, the required images are pulled from your configured registry. When you use the docker push command, your image is pushed to your configured registry.

- **Docker objects**
  - When you use Docker, you are creating and using images, containers, networks, volumes, plugins, and other objects. This section is a brief overview of some of those objects.

Docker Images

- An *image* is a read-only template with instructions for creating a Docker container. Often, an image is *based on* another image, with some additional customization.

- For example, you may build an image which is based on the ubuntu image, but installs the Apache web server and your application, as well as the configuration details needed to make your application run.

Docker Images

- You might create your own images or you might only use those created by others and published in a registry. To build your own image, you create a *Dockerfile* with a simple syntax for defining the steps needed to create the image and run it.

- Each instruction in a Dockerfile creates a layer in the image. When you change the Dockerfile and rebuild the image, only those layers which have changed are rebuilt. This is part of what makes images so lightweight, small, and fast, when compared to other virtualization technologies.

Containers

- A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI.

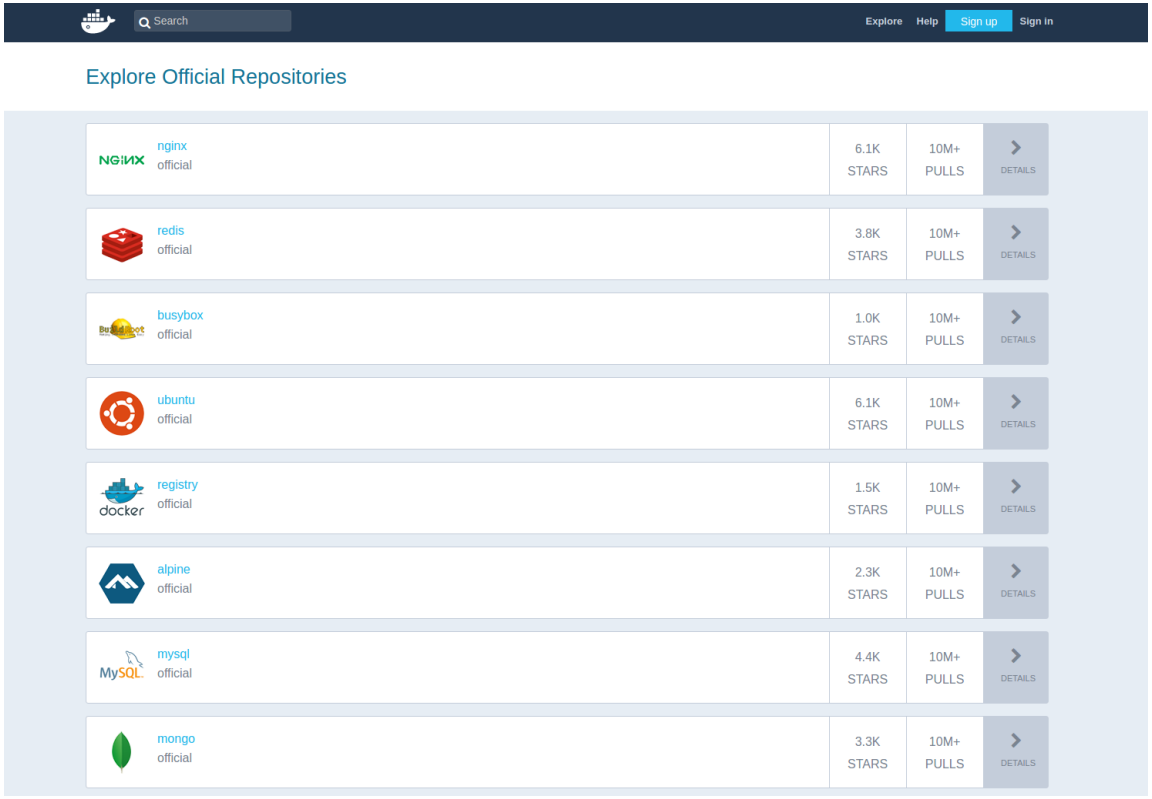
- You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.

Containers

- By default, a container is relatively well isolated from other containers and its host machine. You can control how isolated a container's network, storage, or other underlying subsystems are from other containers or from the host machine.

- A container is defined by its image as well as any configuration options you provide to it when you create or start it. When a container is removed, any changes to its state that are not stored in persistent storage disappear.

Docker Images



E-Resources

- 1. Thomas Erl, ZaighamMahmood and Ricardo Puttini, Cloud Computing: Concepts, Technology & Architecture, Pearson, ISBN :978 9332535923, 9332535922, 1 st Edition

- 2. Anthony T. Velte Toby J. Velte, Robert Elsenpeter, “Cloud Computing: A Practical Approach”, 2010, The McGraw-Hill.