CHP 1:
Introduction to Cloud Computing:

- Cloud Computing (CC): Refers to the delivery of computing resources, such as servers, storage, databases, software, and more, over the internet. It provides on-demand access to these resources without the need for local infrastructure or maintenance.

- Comparing CC with Virtualization: Virtualization involves the creation of virtual versions of physical resources, such as servers or operating systems, to maximize resource utilization. Cloud computing, on the other hand, focuses on providing scalable and flexible services over the internet.

- Grids: Grid computing connects multiple computers to solve large-scale computational problems. While similar to cloud computing, grids typically require a higher degree of coordination between resources and are often used for specific scientific or research purposes.

- Utility Computing: Utility computing is a model where computing resources are provided as metered services, similar to how utilities like electricity or water are billed. It allows users to pay for resources on a usage basis, making it cost-effective and scalable.

- Client-Server Model: The client-server model is a networking architecture where client devices (e.g., laptops, smartphones) request services or resources from central servers. Cloud computing often follows this model, where clients access services or applications hosted on remote servers.

- P2P (Peer-to-Peer) Computing: P2P computing involves the sharing of resources or services directly between individual devices or nodes. It differs from cloud computing, where resources are centralized and accessed over the internet through a service provider.

- Impact of CC on Business: Cloud computing has transformed how businesses operate by providing scalable infrastructure, cost savings, increased collaboration, improved accessibility, and rapid deployment of services. It enables organizations to focus on their core competencies and reduce the burden of managing complex IT infrastructure.

- Key Drivers for Cloud Computing: The main drivers for cloud computing adoption include cost reduction, scalability, flexibility, disaster recovery, increased collaboration, rapid deployment, and access to advanced technologies without heavy upfront investments.

- Cloud Computing Service Delivery Models: Cloud computing offers three primary service delivery models:

  1. Infrastructure as a Service (IaaS): Provides virtualized computing resources, such as virtual machines, storage, and networks, allowing users to build their own applications or services on top of the infrastructure.

  2. Platform as a Service (PaaS): Offers a complete development and deployment platform, including infrastructure, runtime environment, and development tools. Users can focus on building and running applications without worrying about underlying infrastructure management.

  3. Software as a Service (SaaS): Delivers software applications over the internet on a subscription basis. Users can access and use the software directly without worrying about maintenance or infrastructure.

- Cloud Types: Cloud computing can be categorized into three main types:

1. Private Cloud: Resources and services are dedicated to a single organization and not shared with other entities. Private clouds provide increased control, security, and customization but require higher upfront costs.

2. Public Cloud: Resources and services are available to the general public over the internet. Public clouds offer scalability, cost-efficiency, and reduced maintenance but may have lower control and security compared to private clouds.

3. Hybrid Cloud: Combines both private and public clouds, allowing organizations to leverage the benefits of each. It enables seamless data and workload portability between different environments, providing flexibility and scalability.

- When to Avoid Public Cloud: While public clouds offer numerous advantages, there are situations where it may be advisable to avoid them:

- Regulatory or compliance requirements that necessitate specific data handling or storage practices.
- Highly sensitive data or intellectual property that requires strict control and security.
- Applications with stringent performance and latency requirements that may not be

feasible to achieve in a public cloud environment.

- Cloud API: Cloud Application Programming Interfaces (APIs) are sets of protocols and tools that allow developers to interact with and utilize cloud services. Cloud APIs enable the integration of cloud services into applications, enabling automation, resource management, and data manipulation. They provide functions such as provisioning resources, managing storage, networking, and authentication within the cloud environment.

Note: This cheatsheet provides a brief overview of the mentioned topics. For a comprehensive understanding, it is recommended to refer to additional resources and materials on cloud computing.

CHP 2:
Virtualization:

- Introduction to Virtualization: Virtualization is the process of creating a virtual (software-based) version of a resource or system, such as servers, storage devices, networks, or operating systems. It allows multiple virtual instances to run on a single physical machine, enabling efficient utilization of resources, isolation, and flexibility.

- Benefits of Virtualization:
1. Server Consolidation: Virtualization enables running multiple virtual servers on a single physical server, reducing hardware costs and maximizing resource utilization.
2. Isolation and Security: Virtualization provides strong isolation between virtual instances, preventing interference and enhancing security.
3. Resource Optimization: Virtualization allows dynamic allocation and reallocation of resources, optimizing utilization based on demand.
4. Improved Disaster Recovery: Virtual machines can be easily backed up, replicated, and restored, facilitating efficient disaster recovery processes.
5. Flexibility and Scalability: Virtualization provides the ability to easily scale up or down resources as needed, improving agility and responsiveness.
6. Simplified Management: Centralized management tools make it easier to provision, monitor, and manage virtual resources.

- Implementation Levels of Virtualization:

1. Full Virtualization: In full virtualization, a complete virtual machine (VM) is created, including a virtual operating system (OS) that runs on a virtualization layer called the Virtual Machine Monitor (VMM) or hypervisor. The VMM intercepts and manages hardware requests from the virtual machines, providing them with isolated and independent execution environments.

2. Para-virtualization: Para-virtualization is similar to full virtualization, but the virtual machines are aware of the virtualization layer. Guest operating systems are modified to interact with the hypervisor directly, resulting in improved performance compared to full virtualization.

3. Hardware-assisted Virtualization: Also known as native or bare-metal virtualization, this approach utilizes hardware features, such as Intel VT-x or AMD-V, to enhance virtualization performance. It enables the virtual machines to run with minimal interference from the hypervisor.

- VMM Design Requirements and Providers: The Virtual Machine Monitor (VMM) or hypervisor is responsible for managing and coordinating the virtual machines. VMM design requirements include:
1. Isolation: Ensuring strong isolation between virtual machines to prevent interference and maintain security.
2. Resource Allocation: Efficiently allocating and managing physical resources among virtual machines.
3. Performance: Minimizing overhead and providing near-native performance to virtual machines.
4. Compatibility: Supporting a wide range of operating systems and applications.
5. Manageability: Offering tools for provisioning, monitoring, and managing virtual machines.

Examples of popular VMM providers include VMware with their VMware ESXi, Microsoft with Hyper-V, KVM (Kernel-based Virtual Machine), and Xen.

- Virtualization at OS Level: Operating System (OS)-level virtualization, also known as containerization, allows multiple isolated user-space instances, known as containers, to run on a single OS kernel. Each container shares the host OS resources, but appears as a separate entity with its own filesystem, processes, and network interfaces. Benefits of OS-level virtualization include:
1. Efficiency: Containerized applications have lower overhead compared to full virtualization as they share the host OS kernel.
2. Fast Deployment: Containers can be started and stopped quickly, enabling rapid application deployment and scaling.
3. Scalability: Containers can be easily replicated and distributed across multiple hosts to scale applications.
4. Isolation: Containers provide isolation between applications, preventing conflicts

and ensuring security.
5. Portability: Containers are portable across different environments, allowing easy migration and deployment.

Popular OS-level virtualization technologies include Docker and Kubernetes.

Note: This cheatsheet provides a brief overview of virtualization. For a comprehensive understanding, it is recommended to refer to additional resources and materials on virtualization.

Middleware support for Virtualization:

- Middleware: Middleware refers to software components that provide services and functionality between the operating system and applications. In the context of virtualization, middleware plays a crucial role in enabling efficient and seamless virtualization.

- Virtualization Structure/Tools and Mechanisms:
  1. Hypervisor: Also known as a Virtual Machine Monitor (VMM), a hypervisor is a software layer that allows the creation and management of virtual machines (VMs) on a physical host machine. It provides the necessary abstractions and controls for running multiple operating systems simultaneously.
  2. Xen Architecture: Xen is an open-source hypervisor that supports paravirtualization and hardware-assisted virtualization. It follows a microkernel design, where the hypervisor runs directly on the hardware and manages the guest operating systems.
  3. Binary Translation with Full Virtualization: In full virtualization, the hypervisor translates guest instructions (binary translation) to ensure compatibility with the underlying hardware. It allows unmodified operating systems to run on the virtual machines.
  4. Para-virtualization with Compiler Support: In para-virtualization, the guest operating systems are modified to interact with the hypervisor directly. Compiler support is utilized to modify the guest OS kernel and applications to make hypercalls instead of using hardware instructions.

- Virtualization of CPU, Memory, and I/O Devices:
  1. CPU Virtualization: CPU virtualization enables multiple virtual machines to share and utilize the physical CPU resources. The hypervisor manages CPU scheduling, resource allocation, and provides an abstraction layer to ensure isolation and performance.
  2. Memory Virtualization: Memory virtualization allows efficient sharing and allocation of physical memory among virtual machines. The hypervisor manages memory mapping, allocation, and ensures isolation between virtual machines.
  3. I/O Virtualization: I/O virtualization enables virtual machines to share and access physical I/O devices, such as network adapters or storage controllers. The hypervisor provides virtual device drivers and manages device access and resource allocation.

- Hardware Support for Virtualization in Intel x86 Processor:
  The Intel x86 processor architecture provides hardware-level features and extensions to enhance virtualization performance and capabilities. Some key features include:
  1. Intel VT-x (Virtualization Technology): Intel VT-x provides hardware support for CPU virtualization, including virtual machine entry/exit, memory management, and privileged instructions handling.
  2. Extended Page Tables (EPT): EPT enhances memory virtualization by allowing the hypervisor to manage virtual-to-physical address translation efficiently.
  3. Intel VT-d (Virtualization Technology for Directed I/O): Intel VT-d provides hardware support for I/O virtualization, enabling direct assignment of I/O devices to virtual machines and improving device isolation and performance.

- Virtualization in Multicore Processors:
  Virtualization in multicore processors involves leveraging the capabilities of multiple processor cores to enhance virtualization performance and scalability.
  1. Processor Core Partitioning: Multiple virtual machines can be allocated to different processor cores, allowing parallel execution and improved performance.
  2. CPU Scheduling: The hypervisor utilizes advanced scheduling algorithms to distribute the workload across processor cores efficiently.
  3. Resource Sharing and Isolation: Multicore processors enable better resource sharing and isolation between virtual machines, ensuring fair allocation and preventing interference.

Note: This cheatsheet provides a brief overview of virtualization middleware and related topics. For a comprehensive understanding, it is recommended to refer to additional resources and materials on virtualization and specific virtualization technologies.

CHP 3:
Cloud Computing Services:

- XaaS: XaaS (Everything as a Service) is a collective term that encompasses various cloud computing services. It refers to the delivery of different IT resources and functionalities as services over the internet.

- IaaS (Infrastructure as a Service): IaaS provides virtualized computing resources, including servers, storage, and networking, over the internet. Users have control over the operating system and applications they want to run on the infrastructure.

- PaaS (Platform as a Service): PaaS offers a complete development and deployment platform, including infrastructure, runtime environment, and development tools. It allows developers to focus on building applications without worrying about underlying infrastructure management.

- Leveraging PaaS for Productivity: PaaS platforms provide a range of productivity benefits:
  - Streamlined Development: PaaS simplifies the development process by providing pre-configured environments, tools, and frameworks.
  - Scalability and Flexibility: PaaS platforms offer automatic scaling and flexible resource allocation, allowing applications to handle varying workloads.
  - Collaboration and Integration: PaaS supports collaboration among development teams and offers integrations with various services and APIs.
  - Rapid Deployment: PaaS enables quick and seamless application deployment, reducing time-to-market.

- Services Languages for PaaS: PaaS platforms support multiple programming languages, including:
  - Java: Widely used for enterprise applications and web development.
  - .NET: Popular for building Windows-based applications.
  - Python: Known for its simplicity and versatility.
  - Node.js: Suitable for building scalable and event-driven applications.
  - Ruby: Often used with the Ruby on Rails framework for web development.

- DBaaS (Database as a Service): DBaaS provides managed database services over the internet. It handles database management tasks such as provisioning, backup, replication, and scaling, allowing users to focus on data management and application development.

- SaaS (Software as a Service): SaaS delivers software applications over the internet on a subscription basis. Users can access and use the software directly without worrying about installation, maintenance, or infrastructure.

- Comparison of Various Cloud Computing Providers/Software:
  - Amazon Web Services (AWS): Offers a comprehensive suite of cloud services and has a vast user base.
  - Microsoft Azure: Provides a wide range of cloud services, with strong integration with other Microsoft tools and technologies.
  - Google Cloud Platform (GCP): Offers various cloud services and is known for its data analytics and machine learning capabilities.
  - IBM Cloud: Provides a range of cloud services, including AI, blockchain, and IoT.
  - Oracle Cloud: Offers a suite of cloud services, particularly targeting enterprise customers.
  - Salesforce: Known for its customer relationship management (CRM) software delivered as a service.
  - Dropbox: Provides cloud storage and file-sharing services.
  - Slack: Offers team collaboration and communication tools.
  - Zoom: A popular video conferencing and communication platform.
  - Adobe Creative Cloud: Delivers a suite of creative software for design, video editing, and more.

Note: This cheatsheet provides a brief overview of cloud computing services. It's important to conduct further research and consider specific requirements before choosing a cloud computing provider or software.

CHP 4:
Open Source Cloud Implementation and Administration:

- OpenStack Architecture:
  - Features: OpenStack is an open-source cloud computing platform with scalability, multi-tenancy, resource abstraction, automation, and modularity as its key features.
  - Components: OpenStack consists of core components such as Nova, Neutron, Cinder, Swift, Keystone, Glance, Horizon, Heat, Ceilometer, and Trove, each responsible for specific cloud functionalities.
  - Modes of Operations: OpenStack can be deployed in All-in-One (AIO), Proof of Concept (PoC), High Availability (HA), or Distributed modes based on the deployment requirements.

- Installation and Configuration:
  - Choose a deployment tool such as DevStack, Packstack, or OpenStack-Ansible.
  - Follow the installation guide provided by the chosen deployment tool to set up and configure OpenStack.
  - Configure networking, storage, and other components based on the deployment architecture and requirements.

- Cloud Administration and Management Tasks:
  - User and Project Management: Create and manage user accounts and projects (tenants) within OpenStack.
  - Resource Allocation and Quotas: Configure resource quotas and manage resource allocation for projects.
  - Networking Configuration: Set up and manage networks, subnets, routers, and security groups.
  - Storage Management: Create and manage storage volumes, object storage containers, and backups.
  - Monitoring and Logging: Monitor resource usage, performance, and generate logs for troubleshooting and auditing.
  - Security and Access Control: Configure security measures, access control policies, and authentication mechanisms.

- Creating User Interface (Web Interface) of Private Cloud:
  - OpenStack provides a web-based dashboard called Horizon for managing and accessing OpenStack services.
  - Customize the Horizon UI to match the branding and specific requirements of the private cloud deployment.
  - Use Horizon to perform management tasks, provision resources, monitor usage, and access project-specific settings.

- Open Source Cloud Alternatives:
  - Apache CloudStack: Open-source cloud computing software for creating, managing, and deploying infrastructure cloud services.
  - Kubernetes: An open-source container orchestration platform that can be used to build and manage cloud-native applications.
  - OpenNebula: Open-source cloud management platform for managing virtualized data centers and hybrid cloud environments.
  - Eucalyptus: An open-source private cloud software platform compatible with Amazon Web Services (AWS) APIs.

Note: This cheatsheet provides a brief overview of implementing and administering an open-source cloud using OpenStack. It is recommended to refer to the official documentation and additional resources for detailed instructions and best practices.

CHP 5:
Cloud Deployment:

- Factors for Successful Cloud Deployment:
  1. Planning and Strategy: Develop a clear cloud strategy and roadmap aligned with business objectives.
  2. Resource Assessment: Evaluate resource requirements, scalability needs, and workload characteristics.
  3. Provider Selection: Choose a reliable and reputable cloud service provider based on specific requirements.
  4. Security and Compliance: Implement robust security measures and ensure compliance with relevant regulations.
  5. Data Migration and Integration: Plan and execute smooth data migration and integration with existing systems.
  6. Performance Optimization: Optimize cloud resources, network connectivity, and application performance.
  7. Monitoring and Management: Deploy comprehensive monitoring and management tools to ensure efficient cloud operations.
  8. Training and Skill Development: Provide training and upskill employees to effectively utilize and manage cloud resources.

- Network Techniques Requirements:
  1. High Bandwidth: Ensure sufficient network bandwidth to handle the volume of data and traffic between cloud resources and users.
  2. Low Latency: Minimize latency to reduce response time and improve application performance.
  3. Reliability and Redundancy: Implement redundant network paths and backup connectivity options to ensure high availability.
  4. Security: Deploy robust network security measures, such as firewalls, VPNs, and intrusion detection systems, to protect data and resources.
  5. Scalability: Design the network to scale seamlessly as cloud resources and user demand increase.
  6. Quality of Service (QoS): Prioritize critical traffic and ensure consistent network performance for time-sensitive applications.

- Potential Problem Areas in a Cloud Network and Their Mitigation:
  1. Network Congestion: Mitigate congestion by implementing traffic shaping, load balancing, and quality of service techniques.
  2. Network Security Vulnerabilities: Address security vulnerabilities through proper access controls, encryption, and regular security audits.
  3. Network Outages: Implement redundant network connections and backup options to minimize the impact of network outages.
  4. Data Privacy and Compliance: Adhere to data privacy regulations and implement strong encryption and access controls to protect sensitive data.
  5. Service Provider Reliability: Choose a reliable service provider with a strong track record and SLAs that meet business requirements.

- Cloud Network Topologies:
  1. Public Cloud: Resources and services are provided over the public internet, accessible to multiple clients or organizations.
  2. Private Cloud: Resources and services are dedicated to a single organization, providing greater control and security.
  3. Hybrid Cloud: Combines public and private cloud infrastructure, allowing seamless integration and workload portability.
  4. Multi-Cloud: Utilizes multiple cloud service providers for different services or regions, providing flexibility and avoiding vendor lock-in.

- Automation and Self-Service Features in a Cloud:
  - Automation: Automate routine tasks and processes, such as resource provisioning, scaling, and configuration management, to improve efficiency and reduce manual effort.
  - Self-Service: Provide users with self-service portals and APIs to request and manage cloud resources, enabling quick and on-demand access to services.

- Cloud Performance:
  - Key factors affecting cloud performance include network latency, resource allocation, storage performance, and application design.
  - Performance can be optimized through proper resource provisioning, load balancing, caching, content delivery networks (CDNs), and performance monitoring and tuning.

Note: This cheatsheet provides a brief overview of cloud deployment considerations. It is important to conduct further research and consider specific requirements and best practices when deploying a cloud infrastructure.

CHP 6:
Security:

- Security for Virtualization Platform:
  - Host Security: Implement strong host security measures, such as regular patching, access controls, intrusion detection systems, and monitoring, to protect the virtualization platform from unauthorized access and attacks.
  - SaaS (Software as a Service) Security: Ensure secure access controls, authentication mechanisms, and data protection measures are in place to protect user data and applications within the SaaS environment.
  - PaaS (Platform as a Service) Security: Implement proper access controls, secure APIs, and isolation measures to protect the platform and applications running on it from unauthorized access and data breaches.
  - IaaS (Infrastructure as a Service) Security: Secure the underlying infrastructure by implementing proper network security controls, virtual machine security, and isolation mechanisms to protect customer data and workloads.

- Data Security:
  - Data Security Concerns: Understand and address potential data security concerns, such as unauthorized access, data breaches, data loss, and insider threats.
  - Data Confidentiality and Encryption: Implement encryption mechanisms, both in transit and at rest, to protect sensitive data from unauthorized access. Use strong encryption algorithms and key management practices.
  - Data Availability: Implement redundant storage systems, backup mechanisms, and disaster recovery plans to ensure data availability in case of system failures or disasters.
  - Data Integrity: Implement integrity checks, such as checksums or digital signatures, to ensure data integrity and detect any unauthorized modifications or tampering.
  - Cloud Storage Gateways: Use cloud storage gateways to securely connect on-premises data storage systems with cloud storage, enabling secure data transfer and encryption.
  - Cloud Firewall: Implement cloud firewalls and network security controls to protect against unauthorized network access, malicious traffic, and distributed denial-of-service (DDoS) attacks.

Note: This cheatsheet provides a brief overview of security considerations for virtualization platforms, SaaS, PaaS, IaaS, and data security in the cloud. It is important to conduct further research and consider specific security requirements and best practices when implementing security measures.

CHP 7:

Architecture for Cloud Applications:

- Cloud Application Requirements:
  - Scalability: Ability to handle increasing workloads and accommodate growing user demand.
  - Availability: Ensuring applications are accessible and operational with minimal downtime.
  - Elasticity: Ability to dynamically scale resources up or down based on demand.
  - Resiliency: Ability to recover quickly from failures and maintain service continuity.
  - Security: Implementing robust security measures to protect data and ensure user privacy.
  - Performance: Optimizing application performance and responsiveness.
  - Interoperability: Integrating with other cloud services and APIs for seamless operation.

- Architecture for Traditional Cloud Applications vs. Cloud-Native Applications:
  - Traditional Cloud Applications: These are applications initially designed for on-premises environments but later migrated to the cloud. They may not fully leverage cloud-native features and scalability.
  - Cloud-Native Applications: These applications are designed specifically for cloud environments, utilizing cloud-native services and scalability features. They are built with microservices, containers, and serverless computing.

- Multi-Tier Application Architecture:
  - Presentation Tier: Handles the user interface and user interactions.
  - Application (Logic) Tier: Contains the business logic and application processing.
  - Data Tier: Manages the storage and retrieval of data.

- Service-Oriented Architecture (SOA) for Cloud Applications:
  - Resource-Oriented SOA: Focuses on exposing resources as web services, allowing clients to interact with these resources through standard protocols like HTTP.
  - Method-Oriented SOA: Emphasizes service interfaces and exposes methods or operations to clients, providing specific functionality.
  - Event-Driven SOA: Utilizes events and messages for communication and coordination between services, allowing asynchronous and loosely coupled interactions.

- Parallelization within Cloud Applications:
  - Cloud applications can leverage parallel processing techniques, such as distributed computing, to perform tasks concurrently across multiple resources, improving performance and scalability.
  - Techniques like data partitioning, workload distribution, and parallel algorithms can be used to parallelize processing within a cloud application.

- Leveraging In-Memory Operations for Cloud Applications:
  - In-memory operations involve storing and processing data in memory instead of accessing disk storage, resulting in faster processing and improved performance.
  - Cloud applications can leverage in-memory databases, caching, and data grids to accelerate data-intensive operations and enhance responsiveness.

Note: This cheatsheet provides a brief overview of cloud application architecture considerations. It is important to conduct further research and consider specific application requirements, best practices, and architectural patterns when designing cloud applications.

CHP 8:
Adoption and Use of Cloud:

- Adoption of Public Cloud by SMBs:

- Small and Medium-sized Businesses (SMBs) often adopt public cloud services due to their cost-effectiveness, scalability, and reduced maintenance burden.
  - Public cloud adoption enables SMBs to access enterprise-grade infrastructure and services without heavy upfront investments.

- Cloud Phases for SMBs:
  - Phase 1: Testing and Development: SMBs begin by migrating non-critical applications or development environments to the cloud for testing and development purposes.
  - Phase 2: Production Workloads: As confidence in cloud services grows, SMBs start migrating production workloads to the cloud, leveraging its scalability and flexibility.

- Vendor Liability and Management:
  - Cloud service providers typically offer Service Level Agreements (SLAs) that define their responsibilities, liabilities, and uptime guarantees.
  - SMBs should carefully review SLAs and consider factors such as data backup, security, compliance, and support when choosing a cloud vendor.

- Adoption Process of Public Clouds by Enterprises:
  - Enterprises adopt public clouds for reasons like scalability, cost optimization, agility, and access to advanced services and technologies.
  - The adoption process involves assessing business requirements, identifying suitable workloads for migration, selecting the right cloud provider, and developing a migration strategy.

- Managed Private Clouds:
  - Enterprises may opt for managed private clouds, which provide the benefits of cloud computing while offering increased control, security, and customization.
  - Managed private clouds are typically hosted and managed by third-party providers, offering a dedicated environment tailored to specific enterprise needs.

- Migrating Applications to the Cloud:
  - Impact of Shared Resources and Multi-Tenancy: Applications must be designed to function efficiently in a shared resource environment, ensuring they can handle resource contention and maintain performance levels.
  - Phases during Migration to an IaaS Cloud:
    1. Assessment and Planning: Evaluate application suitability, identify dependencies, and plan the migration strategy.
    2. Replication and Testing: Replicate the application and test its functionality in the cloud environment, ensuring compatibility and performance.
    3. Data Migration: Transfer data to the cloud, ensuring data integrity and security during the migration process.
    4. Deployment and Cutover: Deploy the application in the cloud, perform final testing, and transition users to the new environment.

Note: This cheatsheet provides a brief overview of the adoption and use of cloud services. It's important to consider specific business requirements and consult with experts when planning cloud adoption or migration strategies.

Cloud Programming:

- Programming Support for Google App Engine:
  - Google File System (GFS): A scalable distributed file system designed for Google's infrastructure, providing reliable storage for App Engine applications.

- Bigtable: A distributed, highly scalable, and structured storage system used by App Engine for storing large amounts of structured data.
- Google's NoSQL System: App Engine provides a NoSQL data store for storing and retrieving unstructured data, offering high scalability and flexibility.
- Chubby: A lock service used by App Engine for distributed coordination and synchronization of processes.
- Google Distributed Lock Service: A service that allows distributed applications to coordinate and manage locks across multiple machines.

- Programming Support for Amazon EC2:
- Amazon S3 (Simple Storage Service): A scalable object storage service that provides secure and durable storage for various types of data, accessible via APIs.
- EBS (Elastic Block Store): A block-level storage service that provides persistent storage volumes for EC2 instances, offering durability and low-latency access.
- SimpleDB: A highly available and scalable NoSQL database service provided by Amazon, suitable for storing structured data.

Note: This cheatsheet provides an overview of the programming support offered by Google App Engine and Amazon EC2. It's important to refer to the respective documentation and resources for detailed programming guidelines and best practices when working with these cloud platforms.