1.

The Common Open Policy Service (COPS) Protocol is part of the internet protocol family as described by the Internet Engineering Task Force's RFC 2748. COPS defines a straightforward client/server paradigm for implementing policy control over Quality of Service (QoS) signaling protocols (e.g. RSVP). Policies are maintained on servers, acted on by Policy Decision Points (PDP), and enforced on clients (also known as Policy Enforcement Points) (PEP).

Motivation for developing COPS:

The IETF began to handle the issue gradually, exclusively on a need-to-know basis. The first such requirement was for policy setting to support QoS. The proposed paradigm created additional issues, such as the requirement to keep the network management and the device in sync. Another issue was the possibility of conflict between two or more network administrators who were in charge of the same equipment. Then there's the issue of policy-based management.

Network providers want a system that would allow them to give a resource based on a set of policy rules. The decision to give or deny the resource is based on information about the user, the requested service, and the network itself. Using SNMP for this purpose was difficult, therefore the IETF created a new protocol for interactions between network elements and the Policy Decision Point (PDP), where policy-based choices were made. The protocol is known as the Common Open Policy Service (COPS).

Goals of COPS Framework:

i. Policy-based control over QoS admission control choices, with a major focus on the RSVP protocol. It also supports pre-emption, different policy types, monitoring, and accounting. Pre-emption in this context refers to the capacity to delete a previously provided resource in order to meet a fresh request.

ii. The protocol operates on a client/server architecture, with the PEP sending requests, updates, and deletions to the distant PDP and the PDP returning choices to the PEP.

iii. Authentication, replay protection, and message integrity are all provided by COPS at the message level. To authenticate and protect the channel between the PEP and the PDP, COPS can alternatively employ existing security protocols like IPSec (Internet Protocol Security) or TLS (Transport Layer Security).

iv. For a stable exchange of messages between policy clients and a server, the protocol leverages TCP as its transport protocol. As a result, no additional techniques are required for a server and its clients to communicate reliably.

References: https://datatracker.ietf.org/doc/html/rfc2748
https://en.wikipedia.org/wiki/Common_Open_Policy_Service

2.

COPS is distinct from SNMP or CMIP in that it utilizes a stateful client-server approach, which differs from the remote procedure call. The PEP (client) makes queries to the distant PDP (server), and the PDP answers with decisions, as in any client-server approach. However, unless the PEP expressly deletes them, all requests from the client PEP remain installed and kept by the distant PDP. The choices may take the shape of a succession of notifications in response to a single request. This, in reality, creates a new behavior: two identical requests may result in different answers since the system states when the first and second of these requests come may change depending on which states were implemented. Another interesting aspect of COPS is that PDP may "push" configuration information to the client and then withdraw it.

COPS, unlike SNMP, was designed to exploit self-identifying objects and is hence expandable. COPS is likewise based on TCP, which assures dependable transport. Although COPS uses TLS, it also includes its own techniques for authentication, replay protection, and message integrity.

Resources: Cloud Computing: Business Trends and Technologies

3.

a.
When compared to the implementation of commands in a command-line interface interpreter, the SNMP transactional model and protocol limitations make MIBs more difficult to develop. A logical MIB operation can become a series of SNMP contacts in which the implementation must preserve state until the operation is completed or a failure is determined. A strong implementation must be clever enough to roll the device back into a consistent state in the event of a failure.

b.
Configuration retrieval and playback are not simple with SNMP. One of the issues is that identifying configuration objects is difficult. Another issue is that the naming scheme is quite particular, which means that physical device reconfigurations can disrupt the ability to replay a previous configuration.

c.
The operators were asked to identify needs that had not been adequately addressed during the breakout session. The results of the breakout session were later considered, and the following list of operator requirements was created.

i. From the perspective of the operators, ease of use is a critical criterion for any network management solution.

ii. A clear separation must be made between configuration data, data that reflects operational status, and statistics. Some devices make it difficult to tell which parameters were specified administratively and which were received through other means, such as routing protocols.

iii. The ability to retrieve configuration data, operational state data, and statistics from devices separately, as well as compare them between devices, is necessary.

iv. It is required to allow operators to focus on network setup as a whole rather than individual devices.

v. The ability to perform configuration transactions across several devices will greatly simplify network configuration administration.

vi. Given configurations A and B, it should be able to produce the operations required to get from A to B with minimal state changes and network and system consequences. It is critical for reducing the impact of configuration modifications.

vii. A system for dumping and restoring settings is a basic activity required by operators. It is preferable to have standards for pulling and pushing configurations from/to devices.

viii. It must be simple to perform consistency checks on configurations across time and between the endpoints of a connection in order to detect the differences between two configurations and whether they are consistent.

ix. Network-wide configurations are often kept in central master databases and converted into formats that may be delivered to devices, either by creating CLI commands or entire configuration files. Although the models utilized by various operators are likely relatively similar, there is no universal database format for network configuration.
The common portions of these network-wide configuration database schemas should be extracted, documented, and standardized.

x. It is extremely desirable that text processing tools like diff and version management systems like RCS or CVS be able to handle configurations, which means that devices should not reorder data like access control lists arbitrarily.

xi. The granularity of access control required for management interfaces must correspond to operational requirements. Typical criteria include a role-based access control model and the concept of least privilege, which states that a user should only be granted the minimum amount of access necessary to complete a task.

xii. Access control lists must be checked for consistency across devices.

xiii. It is critical to differentiate between the distribution of configurations and the activation of a specific configuration.
Devices should have the ability to store various settings.

xiv. SNMP access control is data-driven, whereas CLI access control is often command-driven (task-driven). Depending on the management role, data-oriented or task-oriented access control may be preferable. As a result, it is essential to support data-oriented and task-oriented access control.

References: https://datatracker.ietf.org/doc/html/rfc3535
4.

NETCONF does not use REST API in its message layer.

The Messages layer handles the encoding of remote procedure calls (RPCs) and notifications. The obligatory property "message-id" of the rpc> element is a string specified by the sender of the RPC that will often encapsulate a monotonically rising number. This string is neither decoded nor interpreted by the RPC receiver; it is just saved to be used as a "message-id" property in any resulting <rpc-reply> message. The sender MUST make certain that If the sender wishes the string to be returned unchanged, the "message-id" value is normalized according to the XML attribute value normalization rules provided in [W3C.REC-xml-20001006].

Resources: https://en.wikipedia.org/wiki/NETCONF
https://datatracker.ietf.org/doc/html/rfc4741
https://datatracker.ietf.org/doc/html/rfc6241

5.
YANG is the de-facto language for NETCONF, and it is described in IETF 6020 under the category Standard Track, with ISSN 2070-1721.

In YANG, the module is the fundamental unit of specification. A module is responsible for defining a single data model. A module can define a full, coherent model or add nodes to an existing data model. All standard modules and submodules must have unique names. It is RECOMMENDED that enterprise module developers pick names for their modules that have a minimal likelihood of conflicting with standard or other enterprise modules.

YIN Module, on the other hand, is the name of the XML-based representation of YANG.

6.

i. Define the workload — The number and kind of virtual machines required for migration will be determined by the nature and magnitude of the workload, as well as how it interacts with non-migrated applications and services.

ii. Provision of cloud resources — Service providers will give a self-service interface for creating accounts and purchasing the services you require (eg, servers, storage, network).

iii. Create a connectivity bridge — This provides secure and transparent bi-directional connectivity, often over an HTTPS connection. An internet VPN is necessary between your data center and the cloud, both during the transfer and for cross-platform application interactions after the move.

iv. Deploy the workload — Once connectivity is established, virtual machines may be set up and connected to legacy services (such as Active Directory), followed by the transfer of the application and any related databases, software, and services.

v. Ensure seamless two-way access — A smooth interaction between the cloud workload and non-migrated services is necessary, and you must be able to monitor and control both the application and the cloud infrastructure.

vi. Test and validate — Regardless of how thoroughly you prepared and tested before deployment, there may be surprises. Is everything moved properly?

vii. Discontinue the old service — Once you've confirmed that everything is operating properly, you may provide users access and decommission the enterprise service. a

Resources:
https://www.techtarget.com/searchcloudcomputing/tip/Cloud-onboarding-best-practices-for-users-and-applications

7.
The Cloud service provider, Cloud service developer, and Cloud service customer are the three organizations engaged.

Assume that the instances for a load balancer and two servers were successfully built, but the virtual machine for the third server was not. What is the user program supposed to do? For the following reasons, deleting all other instances and restarting is not an expedient course of action. From the standpoint of the service developer, this would significantly complicate the software (which is supposed to be fairly simple). From the perspective of the service provider, this would result in the waste of resources that were assigned, then released, but never used.

Second, after all, instances have been established, a service provider must provide elasticity. The question is how this can be (a) described and (b) implemented. Assume that each of the three servers has attained its maximum CPU use. Then, a simple answer is to establish another instance (which can be destroyed after the surge of activity has ended), but how can all of this be done automatically? To that aim, perhaps just two instances, rather than three, should have been generated in the first place.

The industry's approach is to define a service in more generic terms (we'll explain with examples), such that creating service is an atomic activity executed by the service provider—this is where orchestration enters the picture for the first time. Once the service has been deployed, the orchestrator will add and delete instances (or other resources) as stated in the service specification.

The service provider develops an offering for a service consumer by adding limits, fees, policies, and SLA to this template. When the customer accepts the offer, the consumer and provider engage in a contract that includes, among other things, the SLA and a set of precise, quantifiable characteristics of the SLA known as Service-Level Objectives (SLOs).

Resources: Cloud Computing: Business Trends and Technologies

8.

All OpenStack modules with the word "API" in their name (for example, nova-API) are daemons that provide REST services (as mentioned in the Appendix). The Advanced Message Queuing Protocol is used to communicate amongst daemons (AMQP).
AMQP can begin at either end of the pipe. An HTTP transaction, on the other hand, can only be initiated by the client because HTTP is a client/server protocol.

For basic functionality, it requires the following extra OpenStack services:

i. Keystone: This is responsible for providing identification and authentication for all OpenStack services.

ii Glance: In a nutshell, this is the compute image repository. Glimpse images are used to launch all compute instances.

iii. Neutron: Neutron is in charge of supplying the virtual or physical networks to which compute instances connect when they boot.

iv. Placement: Placement is in charge of keeping track of the inventory of cloud resources and aiding in deciding which provider of those resources will be utilized when constructing a virtual machine.

Resources: Cloud Computing: Business Trends and Technologies
https://docs.openstack.org/nova/latest/