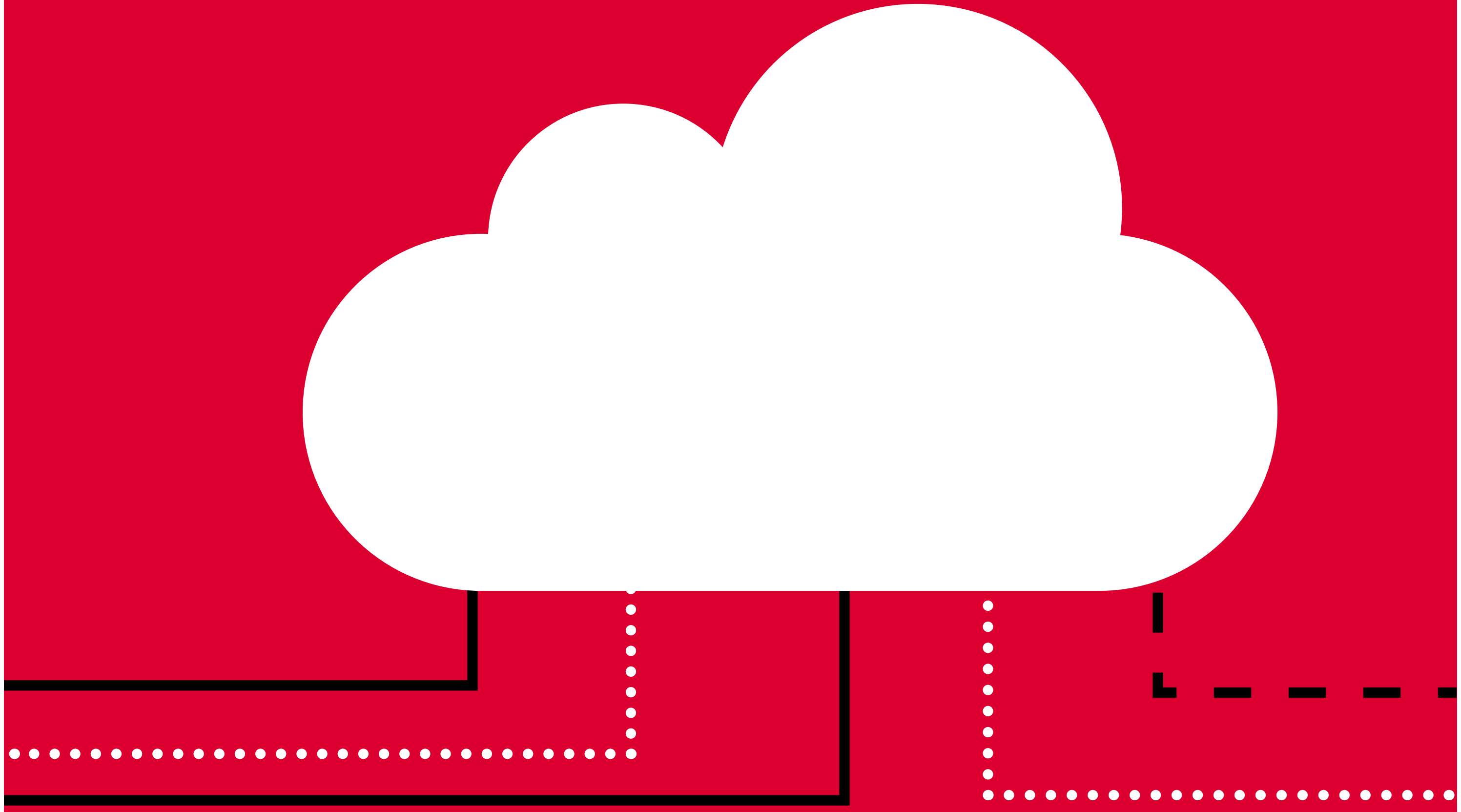
Why The Right APIs Are Super-critical For Your Cloud's Success



Tintri

@tintri **www.tintri.com**

APIs: The foundation of application integration

With the explosion in datacenter apps, APIs have taken on an even more prominent role in enabling their integration. APIs provide a set of communication protocols that, when supported by other applications, allow those applications to interact and share data.

- Clicking a "Like" button for an article published online accesses an API provided by Facebook to add that like to the writer's Facebook page.
- The public cloud is entirely enabled by APIs, providing IT application access to the 3rd-party-hosted resources.
- Virtualization analytics apps leverage the vCenter API to pull hypervisor data and process it in their platform.

Without APIs, all application/resource integration would need to be custom. VMware would have to work with each app developer to create a custom protocol for interoperability, and manually update it every time either software changed.

Previously, when most apps followed a Microsoft Office model of single-provider development, APIs were an ancillary feature. Now, in an age of distributed, independent app development, integration and interoperability have become table stakes for any platform. APIs are the foundation of that ability.

Four Elements of Great APIs

The reason APIs have become critical to application development is because they allow for independent development and massive scale. Yet not all APIs are created equal. Like any software, their design and support can greatly impact their usefulness. For users to get the most out of integrating an API into their application, there are four important characteristics that an API must exhibit.

Simplicity

An API is a means to an end, not an end unto itself. It is a tool for users to leverage to simplify their tasks and expand the functionality of their systems. Users place a premium on intuitive design, a steep learning curve, and effectiveness at delivering on expectations.

A key aspect of supporting these goals is incorporating well-understood standards. Because APIs are a communication medium, leveraging common and well-understood languages, syntax, and heuristics are strongly preferred over anything esoteric or proprietary. Many APIs are adopting REST protocols, which are widely used and well-documented, greatly simplifying management by users. Leveraging these standards provides simplicity and confidence to users and increases the popularity of the API.

In addition to standards, hiding complexity behind a simple and intuitive UI helps users to maximize its value while minimizing its burden. By abstracting away this complexity and presenting only the essentials to the user, the API delivers all required benefits while reducing interactive overhead.

Continuing the virtualization reference above, its whole value proposition is abstracting away the complexity of managing a dispersed and diverse infrastructure. This simplifies user access to resources they need to be successful. By combining standards and abstraction, virtualization experienced breathtaking adoption rates and has transformed all aspects of IT.

Documentation

While APIs should be built on operational simplicity, they should still be backed up by sufficiently extensive documentation. Documentation is key to accelerating the adoption, implementation and management of the API. To get the maximum benefit, users should easily understand all the API's capabilities, debug problems quickly, and optimize its integration into their workflow. According to users, clear, complete and easily navigable documentation is imperative.

Clear documentation of requests and responses, with detailed descriptions of their respective parts, can greatly speed implementation and boost confidence in the API. To bolster this process, examples and tutorials can provide real-world guidance to users. They can also streamline users' ability to incorporate support for the API into their applications.

The ideal combination is simple. First, "getting started" documentation that helps developers get a basic framework up and running. Secondly, more detailed, ideally application-specific, documentation to help users understand the full range of capabilities of the API.

Stability

API customers put considerable effort into building applications and workflows around an API's specification. So APIs should go through as little change as possible, with previous versions maintained for an extended period of time to help users adjust to the change.

If an API is designed to provide a response to a query in a certain format and then that format is changed, applications that already have the original response format integrated in their platform will break. Another problem is when parameter names vary across endpoints. Stability and consistency are critical, as is documenting all changes so applications can be adjusted.

Intuitiveness

Developers don't want to have to learn proprietary commands or operations in order to leverage an API unless absolutely necessary, e.g. there may be no alternative to supporting it in the application. Otherwise, development of that application with support for interfacing and operating with that API could be significantly delayed, which can result in the application being hamstrung or obsolete upon release. Ongoing development and upgrades to the application also become problematic, greatly extending development cycles and increasing market risk.

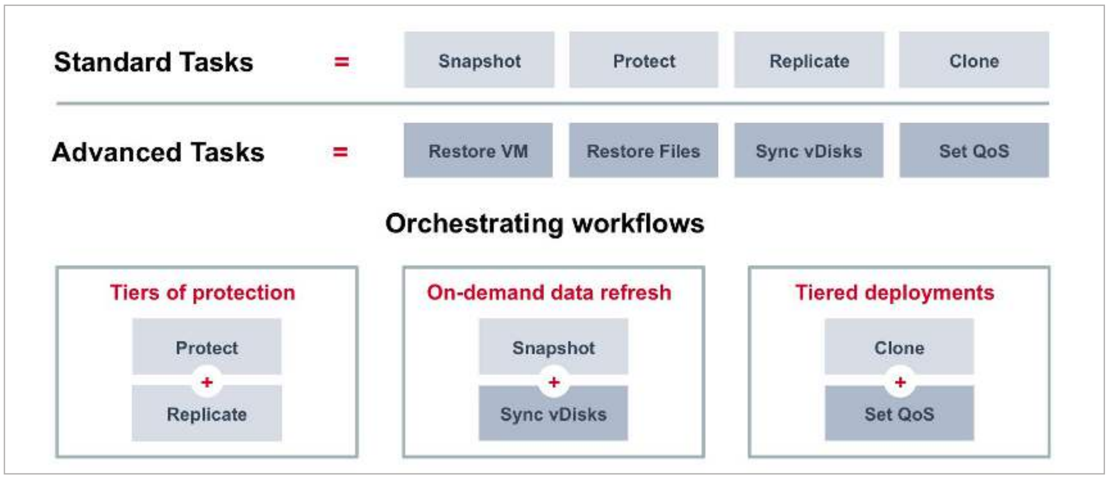
Actual use of an API is greatly simplified with the use of well-understood standard coding and operational methods. An API that is built on JSON for data transfer, REST/HTTP operational protocols and OAuth for authentication leverages universal standards that can significantly lower the barriers to deploying and managing it.

Enterprise applications require careful consideration before moving to cloud, since these applications often benefit from the greater control that on-premises infrastructure provides. If you simply "lift and shift" these apps into the cloud, you may not be entirely happy with the results. It's worth understanding why this is the case (and why it may not always be a fair assumption).

Tintri API: Simplifying and enhancing enterprise cloud and virtual environments

Tintri's API was designed with simplicity in mind. It provides 3rd-party applications with direct access to a broad array of specialized capabilities within the Tintri Enterprise Cloud platform based on Tintri CONNECT architecture. There is no need to switch applications to execute Tintri VM storage management commands. This simplifies and streamlines storage management through the use of standard RESTful commands used for most cloud infrastructure offerings.

By embedding 3rd-party control capability via API, Tintri enables simplified automation of various tasks at the VM level in a workflow. Examples include:



1. Obtaining information on

- Appliance
- Alerts
- Performance statistics
- Snapshots
- VMs
- vDisks
- Datastores
- Licensing
- RBAC
- Replication
- Service groups

2. Automating Standard Data Management Tasks

- Creating Tintri VM snapshots; deletion of snapshots
- Protecting VMs
- Replicate VMs
- Cloning VMs within Tintri environment

3. Automating Advanced Tintri Tasks

- Deleting snapshots
- Restoring VMs
- Restoring files
- Syncing vDisk
- Setting QoS on a VM or service group
- Obtaining VM store information

4. Creating Advanced Orchestrated Workflows

- Combining protection and replication to create tiers of protection
- Combining snapshots and vDisk synch to enable on-demand data refresh
- Cloning and setting QoS for tiered deployments

5. Supporting Advanced Automated Processes

An example of an automated task available through the Tintri API would be retrieving virtual server information en masse. The Tintri Python SDK (PySDK) is a library that provides a Pythonic way to access Tintri APIs. This allows users to leverage a standard language and methodology in order to gather detailed information from a variety of elements in a virtual environment, such as an appliance, datastore, service groups, virtual disk, virtual machine, and VMstore. A specific example can be found at https://tintri.github.io/tintri-python-sdk/build/html/tutorial.html#collecting-server-information

With these powerful capabilities available to a 3rd-party application via API, executing virtualization tasks by using Tintri becomes transparent to the user, who receives seamless, direct access to virtualization aware storage for the cloud infrastructure. No specialized knowledge of Tintri's management dashboard or operational structure is needed. This streamlining of storage management and integration with existing workflows greatly improves productivity and organizational agility without sacrificing the advanced enterprise cloud features designed into Tintri virtualization aware storage.

To support application design that integrates the Tintri API, Tintri has developed a comprehensive set of documentation. You can find all of Tintri's API documentation and examples at https://github.com/Tintri/tintri-rest-api. Legacy API documentation can be found at https://support.tintri.com/.

Tintri understands that stability is a significant factor affecting users' ability to leverage an API. Its API structure is designed to allow for upgrades without causing existing users' applications to break. The Tintri API also supports application stability consistency with how they access the API's capabilities. That supports legacy operations that give time to users to migrate their application operations to any upgraded configurations.

Over time, the Tintri APIs will continue to evolve. Future versions may add new managed entities or keys, but all subsequent versions of the Tintri API will maintain legacy documented behavior and return values with given parameters. Undocumented request parameters or additional response keys, however, may change. For example, when replication capability changes from one destination to many destinations, an array attribute may be added for the many destinations capability, while the old single destination attribute may be maintained and become the first element of the many destinations array.

Finally, Tintri recognizes that developers don't want to burn a lot of time learning how its API operates or how to integrate it into an application. Thus, Tintri built its API on the foundation of common RESTful standards over HTTP with JSON payloads, so developers can quickly understand how it operates and broaden the power of their applications.

Conclusion

If you are trying to optimize your enterprise cloud solution's storage infrastructure supporting applications, you have two choices. You can manage your storage-related virtual infrastructure or enterprise cloud operations via your hypervisor and your storage configurations via each vendor's user interface. Or, you can leverage APIs to handle basic tasks through one platform. Tintri's enterprise cloud solution provides the option to leverage its powerful yet simple API to perform both storage configuration plus all of the common VM configuration operations via one interface. This enables users to streamline enterprise cloud optimization and simplify common VM-related operations.

By giving users the option of working through one platform, Tintri enables them to reduce overhead of their enterprise cloud environment without losing any of its full capabilities. Explore more of the benefits of Tintri Enterprise Cloud at www.tintri.com.