Title: Hosting your services in the cloud - Lessons DuraSpace has learned
Author: Andrew Woods (DuraSpace)

Proposal:
As cloud infrastructure continues to mature, decrease in cost, and increase in usability, repository and I.T. managers, likewise, are compelled to increase adoption. As our community continues down the path of progressively leveraging this infrastructure, certain best practices have begun to emerge. DuraSpace, an early adopter of cloud technologies, both relies on cloud infrastructure for its own internal operations as well as expands upon it in the offering of hosted services. Through the experience of deploying community resources (e.g. wiki.duraspace.org), software as a service (e.g. DuraCloud), and hosted services (e.g. DSpaceDirect) we have both discovered and embraced a set of cloud practices that guide our implementation strategies. This presentation highlights a few of these practices and relates them back to "real world" usage examples in DuraSpace infrastructure and services.

The cloud service provider landscape has evolved significantly over the last five years. We have seen a continual tide of service providers both come and go. Although there are many notable offerings in the cloud services marketplace that provide unique value in their own ways, the clear stand-out among peers is Amazon Web Services (AWS). Amazon has demonstrated through its broad array of capabilities, breadth of market share, and sustained investment that it is the clear leader in providing cloud infrastructure services. Thus, in order to be able to provide concrete examples of usage, this presentation will be framed in the context of AWS features. However, the principles addressed will be applicable to cloud design in general.

There are two primary categories of cloud infrastructure service: compute and storage. Within each category can be found a variety of sub-services that must be understood in their own right. We will begin with a description of cloud compute, then follow with a discussion of cloud storage.

At the outset, it is important to understand that cloud servers (Elastic Compute Cloud, or EC2 instances) are fundamentally different from traditional physical servers. One important difference is the ephemeral nature of server state between restarts. This characteristic is compounded by the fact that as a virtual resource, the lifespan of a given server is not always under the control of the cloud service consumer. There are ways of mitigating this risk, but they require designing the server architecture with this in mind. EC2 instances come with a certain amount of ephemeral storage. Additionally, block storage volumes (Elastic Block Storage, EBS) can be formatted and mounted to an EC2 instance as a

persistent storage device. Along with being persistent, EBS volumes have the added benefit of being elastic in size. That is to say that a volume can be expanded or shrunk as needed. However, simply installing applications and writing state to EBS is not in itself sufficient. It is advisable to create scripted, scheduled backups or snapshots of these EBS volumes. This not only provides rollback points but also protects against the all-too-frequent corruption of EBS volumes. With these basic principles as the foundation, we will now detail some best practices for hosting web applications on cloud compute resources.

The previously mentioned ephemeral nature of cloud compute resources presents a slightly new paradigm for web application deployment. If these compute instances are conceptualized as stateless, dynamically deployable resources, that leads to the possibility of their automatic provisioning and deprovisioning in support of the rise and fall of user load. There are a few enabling ideas that make this possible:

1. The compute resources must be able to be instantiated into an operational state
2. The compute resources must be dynamically added to and removed from a load-balancer
3. The user load on the application must be dynamically monitored

Item number one above is the key design characteristic that web application architects must strive to achieve. The other two items are directly supported by AWS. Applications commonly require configuration and initialization on startup. In order to support this without storing credentials and configuration in the server images, it becomes necessary to employ a secure mechanism for newly instantiated compute resources to call back to a master node for their node-specific setup. There are a variety of ways to achieve this. One common strategy is to make use of a scripting framework such as Puppet or Chef. For DSpaceDirect, we have chosen Puppet to manage server initialization and configuration.

Two keys to enabling highly available web applications and services is to ensure that there is both application redundancy as well as data center redundancy. The general method described above provides for application redundancy by monitoring user load against compute resource status and scaling appropriately behind a load balancer. In terms of data center redundancy, by sharing server images across more than one region AWS offers at the infrastructure-level the ability to duplicate a deployment environment in a completely distinct geographic location. The first key requires designing the service to be horizontally scalable. The second key only involves knowing the option offered by the infrastructure and configuring it.

Another useful and interesting characteristic of cloud computing is the fact that multiple instances of the same server image may be run at the same time and selectively exposed

to Internet users based on an I.P. address mapping. This feature combined with regular, scheduled snapshots of server state is what allows DuraSpace to seamlessly upgrade and restore our hosted community services, such as the wiki, JIRA, source repositories, etc. When an upgrade is needed to the wiki, for example, we spin up a compute resource and attach the previous day's snapshotted EBS volume. Then we perform the upgrade on this secondary compute instance. Once the service has been verified, all that is necessary to make it live is to associate the Elastic I.P. address of the production service to the newly upgraded compute instance. In a similar fashion, if it is determined that the upgrade caused an initially unnoticed negative side-effect, then rolling back to a good state is as simple as re-associating the Elastic I.P. address to the original compute resource.

Having discussed a few capabilities and considerations of cloud compute resources, it is worth introducing the basic cloud storage options in the AWS context. Conceptually, for storage that is persistent within AWS there are three tiers which span the continuum from online and more costly to offline and less costly. The actual cost of the storage tiers depends on the usage scenario, but if each storage option is used in alignment with its intended design then the cost continuum behaves as one would expect. The three storage tiers offered by AWS are Elastic Block Storage (EBS), Simple Storage Service (S3), and Glacier.

Most applications and services hosted in the cloud will take advantage of Amazon's Elastic Block Storage (EBS). This is the highest storage tier, and it acts as online storage space that can be attached to a server. Its advantages are that it can easily be transitioned from one compute resource to another and that it provides fast, mountable data access. Unlike the other two storage options, however, the cost of an EBS volume is based on the allocated volume size as opposed to the amount of storage on the volume that is actually used. So, when you create a 1TB volume, you are paying for 1TB of storage even if you are currently only using 400 GB of it. Additionally, data on EBS volumes can not be accessed directly from web requests, but rather are only available when connected to a cloud server.

Amazon's other two tiers of storage (S3 and Glacier), provide storage which is accessed via the network, rather than through a direct server mount. These options are much more affordable solutions when it comes to archival storage and backup.

S3 is essentially a "nearline", middle storage tier. It is suitable for a variety of use cases as it supports direct HTTP read and write access. As implied above, the cost of S3 storage is exactly proportional to the number of bytes actually used. All access to data stored in S3 occurs through API calls made over the network. Not only does this have performance implications, but it also carries a cost implication since AWS charges for the reading of

bytes over the network. S3 is used by DuraCloud as one of the underlying, web-accessible cloud storage providers along with SDSC, Rackspace, and others.

Finally, the equivalent to offline storage in AWS is the Glacier storage tier. If used as a rarely-read, or potentially dark copy of data, Glacier shines as an extremely cost-effective storage tier. Content stored in Glacier is only available after submitting a request for retrieval. After such a request, the content is accessible three to five hours later. Although not documented, the behavior of Glacier suggests that it is intended to replace existing tape storage systems. Glacier is used by DuraCloud in a similar way to DuraCloud's integration with Chronopolis, as a backup alternative that expects minimal read access.

Given the strengths of the three AWS storage options in different situations, DuraSpace uses each service where it is best suited to maximize utility while minimizing cost.

While this proposal primarily offers a survey of cloud compute and storage services along with related application design implications, the session is intended to not only detail with the architectural impacts of the cloud, but also to more deeply uncover the concrete manifestations of the cloud paradigm within the DuraSpace context. Specifically, the architecture behind DuraCloud as a service, DSpaceDirect, and DuraSpace community resources will be discussed.