QUIXEY

Aliyun Cloud Product

Improvement Proposals

February 20, 2014

Authors of this document
Jaeik Song, Director, Production Operations
Adam Gray, Manager, Dev Platform Engineering
Leigh Klotz, Chief Architect

Contact
Betty Chan (bettychan@quixey.com)

Improvement Proposals Overview

Since our initial engagement with Aliyun has had a year to mature, our gap analysis can be updated with a prioritized point-by-point list of proposed improvements to the Aliyun products.

A major advantage of public cloud platforms and PaaS (Platform-as-a-Service) is that the high degree of API-based provisioning and monitoring allows the traditional personnel and process driven rack-and-stack operations to be treated as a software component, leveraging the benefits of the software engineering lifecycle. “Infrastructure as Code” goes beyond simple automation, and integrates the application, the platform, and the server layers into a single system, providing improved reliability, scalability, and speedy delivery of innovations.

The success of Infrastructure as Code in bringing forth these advantages is limited when the PaaS cloud platform introduces manual steps, unreliable interfaces, or inflexible obstacles. The AliYun cloud product improvement proposals in this document are driven by these concerns, and each can be tied to specific advantages in automation and to specific failures in production systems.

Improvement Proposal Objectives

The thirteen specific changes proposed in this document will improve the Delivery, Reliability, and Scalability of new product releases and features.

Delivery objective

Delivery of new product releases and features is a largely automated task following industry best practices. In AliYun this automation is typically blocked by a lack of basic automation in services, or by a significant difference in feature richness between AliYun and benchmark IAAS providers such as AWS. Quixey expends significant effort compensating for Delivery problems which reduces the level of automation in global deployments. This leads to reduced feature pipeline velocity worldwide. Service provisioning and service information are the major focus of required improvements to provide better delivery of new product releases and features.

Reliability objective

Reliability of deployed services relies on monitoring and automated reactions. Problems in this area typically are the lack of service availability, server or server class availability, and lack of basic automation facilities and features.

Scalability objective

Scalability of deployed services relies on monitoring and automated reactions. Problems in this area typically are the lack of service availability, server or server class availability, and lack of basic automation facilities and features.

Improvement Proposal 1

The ability to use at specify instance creation in the least 3 "available areas" within a region

Business Values Affected

Reliability, Scalability, Delivery

Use Case

It is critical to the reliability of any deployment in a cloud for a user to allocate servers to a group of available zones. In datacenter design terms, these are buildings within a campus or sections of the data center with dedicated service yards. In the cloud, these are generally called availability zones or areas. The uptime of the cloud compute service is defined as properly utilizing all availability zones concurrently, not just spinning up new instances into different zones.

Example: INC-217

Hangzhou availability zone D network failure, confirmed due to municipal construction causing damage to the fiber operators: field undergoing emergency repairs. At present some links have been restored. Please help us test observations and please understand the inconvenience!

Reliability

Reliability was affected because an AliYun Zone experienced a failure and the remaining servers were not able to serve traffic. In order to survive the loss of a zone it must be possible to serve traffic from the other zone or zones.

The AliYun ECS API does not support requesting a server in a zone, so it is not possible to create a deployment footprint with redundancy in different zones. With random allocation of ECS instances to zones, it is not possible to assure that the servers are spread evenly across the zones.

Services such as Kafka, Zookeeper, ElasticSearch, and Cassandra heavily rely on zone isolation in order to provide HA (High Availability). Additionally, some of these services require a majority of servers in a cluster to be up, so there must be at least 3 zones.

Delivery

Delivery is affected because the ECS API does not support requesting a server in a zone, and monitoring software and deployment software cannot assure proper distribution of services.

Requested Action

The ECS API should support a new parameter on the `Action=CreateInstance` call, allowing all users to specify in which zone to create an instance. Three serial API requests with these parameters should not return an error to the user that the zone is unavailable:

1. `Action=CreateInstance&ZoneId=cn-hangzhou-a`
2. `Action=CreateInstance&ZoneId=cn-hangzhou-b`
3. `Action=CreateInstance&ZoneId=cn-hangzhou-c`

Improvement Proposal 2

Local SSD instance store, not on a SAN

Business Values Affected

Scalability, Delivery

Use Case

Aliyun ECS storage is designed with durability first and performance second. We need a way to use high-performance temporary disk storage at the cost of it being non-durable. AWS EC2 implements this as "instance store."

Example: INC-157

YunOS shifted app search traffic away from Quixey because the AliPay Wallet app was not visible for or within 3 hours.

Scalability

Scalability is enhanced because services that need local storage for temporary usage can be more efficient.

Delivery

Delivery is enhanced because global services that assume local storage do not need to be re-written by developers.

Requested Action

All ECS regions should support the "ephemeral" disk type as documented. This ephemeral disk type should be a local SSD which is erased when an instance is stopped or deleted ("released").

Improvement Proposal 3

Instance Tags

Business Values Affected

Delivery, Reliability

Use Case

Instance tags are per-server key-value pairs that are set during the creation of an instance and optionally updated throughout the life of an instance. The provisioning process can use a standard image which itself, during boot, reads the tags to determine the role and parameters of the deployment and finish deployment or configuration.

Additionally, tags can associate resource information such as environment ("staging" vs "production"). The following tags can be used by:

- The host service to control its operation.
- Management services for monitoring and reliability.

Example: INC-229

Tags missing from the host database caused content and code deployments to fail.

Reliability

Reliability is enhanced because instance tagging allows automation to manage clusters and services as opposed to micro-managing individual instances which have a much shorter lifespan than dedicated servers.

Reliability is enhanced because monitoring software can use a single source of truth about the role of each server, its staging-vs-production environment, and its role in a cluster rather than being configured to understand particular host names.

Delivery

Delivery is enhanced because industry best practices can be more easily leveraged with instance tags and a HA service for creating and accessing them.

Requested Action

Replace the Quixey-built hostdb with an instance tagging service that is usable both as part of the AliYun ECS API, and from within the instance by a highly reliable mechanism such as a HTTP service.

As an example, AWS EC2's "metadata service" is available from every instance at http://169.254.169.254/ and documented at http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html

Improvement Proposal 4

Auto-scaling from Self-Published JianKong metrics

Business Values Affected

Scalability, Delivery, Reliability

Use Case

The ESS product only supports a very limited interface for scaling up the group and scaling down the group. Scaling on the basis of internal metrics or per-server metrics is unreliable; best practices call for the use of application-generated metrics, published back to an alerting service for scaling. In a distributed environment, it may not be the slowest service that needs to be scaled in order to handle more traffic through the entire distributed system.

Scalability

Scalability is enhanced because the ECS service can be automatically scaled in response to true business metrics and does not require human intervention. It can be quicker to respond and avoid misses.

Delivery

Delivery is enhanced because an Auto-Scaling facility is widely used and integrated into cluster deployment. For services such as Cassandra, ElasticSearch, Kafka, and Zookeeper, the processes for integrating a new server into a cluster widely vary, and automating them makes it possible to correctly perform the operations without losing traffic or data.

For other cases, such as bringing on new services, the work must be performed many times for developer systems and again for staging and production systems. This increases delivery time and causes a bottleneck in operations for the release of new services.

Reliability

Reliability is affected because without Auto-Scaling metric integration, it is not possible to automate the Auto-Scaling operations. Operations staff must manually scale or replace failed services. As a result, bringing new service components online becomes more expensive because there are more members of a cluster.

Requested Action

AliYun should support the ability to publish custom metrics JianKong which ESS will then act on. This is similar to AWS's integration between AutoScaling and CloudWatch.

Improvement Proposal 5

Guaranteed I/O Rate for Cloud Disk

Business Values Affected

Scalability, Reliability

Use Case

We have noticed some extremely troubling symptoms on the larger instance types and see disk I/O drastically slow down. Network congestion can cause many reliability issues. We suspect that this is due to an instance being co-hosted on an overloaded physical host. We need an option to guarantee a level of throughput to the cloud disk.

Scalability

Scalability is affected because servers are unable to store and quickly retrieve data from the Cloud Disk.

Reliability

Reliability is affected because servers are not performing at consistent levels.

Requested Action

Add an instance configuration option to guarantee specific I/O rates for access to Cloud Disk.

Improvement Proposal 6

Prepaid AliYun services

Business Values Affected

Delivery, Scalability, Reliability

Use Case

Some services such as RDS and OCS require pre-payment using AliPay voucher. The complex integration with financial systems is a barrier to automating the development of testing and production clusters. Prepaid vouchers are also difficult to obtain during urgent outages. The requirement of pre-payment severely cripples our ability to create new infrastructure as new partnership requirements arise.

Delivery

Delivery is impacted because we are unable to quickly create new infrastructure which relies on pre-paid products, forcing us to go through a 24 hour communication cycle to get vouchers allocated and activated.

Scalability

Scalability is impacted because we are unable to automatically add new RDS instances when needed, nor provision more OCS capacity when needed.

Reliability

Reliability is impacted because we are unable to do automated proactive migrations with any of these services, forcing us to react when they fail.

Requested Action

Please make any and all changes to our account, billing, or business arrangement that results in Quixey Engineering staff to never use an AliPay voucher. We need to be able to automate the creation and release of all our AliYun resources.

Improvement Proposal 7

Scheduled API maintenance should not result in API downtime

Business Values Affected

Delivery, Reliability

Use Case

We want to use Aliyun's services to handle our needs as much as possible. To this end, we defer to the Aliyun APIs as authoritative sources of information about our instances and load balancers. For example, our deployment automation requests a list of back-end servers from a SLB before proceeding with any action.

Example: INC-161

Content and code changes failed due to an unplanned service outage during planned maintenance.

Delivery

Delivery is impacted because we refer to the Aliyun ECS and SLB APIs in our deployment automation when releasing a YunOS app store data refresh.

Reliability

Reliability is impacted because when automated processes responsible for system health monitoring and recovery are unable to run, services may operate erratically.

Requested Action

We should not need to pause deployments due to API maintenance. The API maintenance could cause an increased risk of impact, but should not intentionally result in the API being unavailable. When the scheduled maintenance stipulates that read-only operations will continue to function, that guarantee must be honored or acknowledged when broken.

Improvement Proposal 8

Upgraded Private Network Capacity to 1Gbps+

Business Values Affected

Scalability, Reliability, Delivery

Use Case

AliYun does not offer sufficient network capability to handle scaled data clusters of the types needed to effectively support traffic.

Scalability

Scalability is enhanced because we can more efficiently use the resources that are needed rather than over-provision CPUs because we have a severe memory-bound problem (as an example). Also, large data systems like Cassandra and ElasticSearch require massive network throughput to reconcile data during recovery or expansion.

Reliability

Reliability is enhanced because we can more accurately request services that are needed instead of using a single pool of high-performance servers for everything.

Delivery

Delivery is enhanced because we can use a worldwide footprint of server types for similar services and spend less time customizing services code for particular CPU/memory balances. This will enable us to spend less time re-testing distributed computation parameters such as shard sizes.

Requested Action

We would like an instance family with higher network bandwidth of 1Gbps minimum.

Improvement Proposal 9

AliYun API Client SDK for Python

Business Values Affected

Delivery, Scalability, Reliability

Use Case

Automation and treating infrastructure as code relies heavily on the ability of the developers and the developer operations team to smoothly integrate monitoring, deployment, and scaling. We heavily use the python AWS SDK called boto in our deployment automation.

Boto allows us to:

- Easily create new clusters through AutoScaling Groups.
- Update existing instances by managing Elastic Load Balancer membership.
- Automatically give new engineers a day-1 task of creating a personal development instance themselves.

The Aliyun API had no viable Python SDK to meet these expectations, so we had to create one and maintain it.

Example: ALI-14 (shared ticket with Aliyun)

Aliyun ECS removed an instance type without clear notice.

Delivery

Delivery is affected because we are forced to decrease the sophistication of our deployment automation and cluster management to the simplest commonality between our Aliyun API SDK for Python and Boto. We are unable to update our Aliyun API SDK for Python as quickly and frequently as is required for a growing product like Aliyun.

Scalability

Scalability is impacted because our Aliyun API SDK for Python necessarily lags behind new fundamental Aliyun API scaling features like ESS.

Reliability

Reliability is impacted for much the same reason as scalability. If we have a Scaling Group and an instance becomes overloaded, we can publish a metric to ESS to have the instance deleted and automatically replaced.

Requested Action

Provide a higher-level Python SDK than the current one provided by Aliyun along the lines of our project: https://github.com/quixey/python-aliyun.

Quixey would be extremely excited if Aliyun engineers began contributing to this project. This could be a great benefit to Aliyun since we are sure there are existing and potential customers who would benefit from a more sophisticated Python SDK.

Improvement Proposal 10

Granular Access Control

Business Values Affected

Delivery

Use Case

The current Aliyun access control is all-or-nothing for the management console. The RAM service looks promising, but requires a new Aliyun account for each access pattern. Aliyun accounts require a Chinese SMS and AliPay account, which is extremely difficult to do for us.

Having API access keys which are able to use different levels of access would get us most of the way toward proper access control. This would allow us to disable the use of the management console until it supports sub-accounts with different access.

Delivery

Delivery is impacted because we have limited access control of Aliyun resources, forcing us to compromise from industry-standard best practices of principles of least privileges. We therefore have to tightly guard every Aliyun API access key, causing many delays and procedural roadblocks for engineers to create new product features and prototypes in Aliyun.

Requested Action

Support multiple access principals within one Aliyun account. We would like to associate a new set of API keys with specific Aliyun resource access. As a specific example, we want an API key for a "Production Deploy" user to be able to access SLB, ECS and ESS resources which are associated with an access control path of "production".

Furthermore, we want an API key for myself (personally) to be unable to create, read, modify or delete any resources which are associated with an access control path of “production.” This is a small sub-set of AWS Identity and Access Management.

Improvement Proposal 11

Hosted DNS Service

Business Values Affected

Reliability, Delivery

Use Case

Quixey maintains our own internal DNS servers since Aliyun does not currently offer a DNS service that we can use.

Example: INC-205

Quixey's internal DNS servers were pointed to a stale set of Aliyun DNS servers causing API calls to oss-internal.aliyuncs.com to fail.

Reliability

Reliability is impacted since we do not control the upstream DNS servers. When they change, we need to update our own internal servers.

Requested Action

Deploy a DNS service.

Improvement Proposal 12

Service Health Dashboard

Business Values Affected

Reliability, Delivery

Use Case

We have seen many issues with our reliability when accessing OSS. Specifically, we have seen DNS resolution temporarily fail, requests intermittently fail, and short-notice API changes. Our use of OSS requires rapid, frequent requests to its API in order to store and retrieve many large files. When some of these requests fail, large file syncs are compromised and must be re-attempted.

Reliability

Reliability is impacted because we use OSS as a shared network storage resource for copying data between instances, storing built code for deployment, and as a cross-site data transfer point.

Delivery

Delivery is impacted because we use OSS as the intermediate content storage service when exchanging app store data with Alibaba and internally as we process that app store data.

Requested Action

Schedule planned maintenance and quickly announce when unplanned activities are required. Track and report the OSS SLA including clients on the other side of the great firewall, DNS availability of:

- oss.aliyuncs.com
- *.oss.aliyuncs.com
- oss-internal.aliyuncs.com
- *.oss-internal.aliyuncs.com

Improvement Proposal 13

Upgraded RAM Capacity

Business Values Affected

Delivery

Use Case

Quicker time-to-market for new solutions iterations.

Delivery

Delivery is impacted as memory constraints will limit the engineering options that Quixey will be able to use in delivering new solutions.

Requested Action

We would like the ability to allocate at least 8 instances with 128GB of RAM and 16 CPU modern CPU cores.