# Designing & Consuming a Cloud 2.0 API

@jclouds

jclouds

multi-cloud library

## First Things First

Thanks!

QCon

InfoQ

software pilots

TRIFORK.

jclouds

## Agenda?

# Agenda

- Why Cloud APIs Matter
- State of the Cloud
- Why a Cloud 2.0 API?
- Designing a C2API
- Consuming a C2API

jclouds

# Why Cloud APIs Matter

- Connected Ecosystem!
- Delight Users
- Encourage Community
- Your API = Your Service

## State of the Cloud

- HTTP(s) is da lingo
- Beyond that, a mixed bag:
  - Layouts
  - Formats
  - Authentication
  - …
- it's all different!

@jclouds

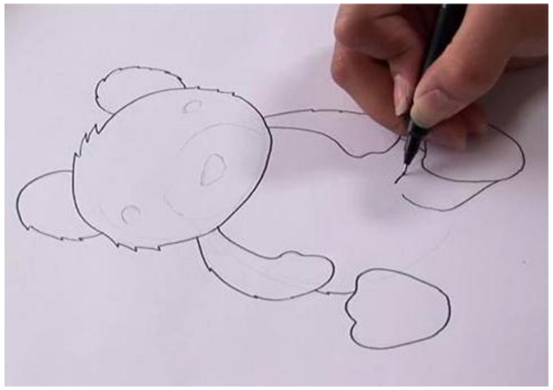
jclouds

## Why a Cloud 2.0 API?

- Deal with some cloud-specific challenges
- Have seen enough attempts to see what works
- Help Your Community

# Designing a Cloud 2.0 API

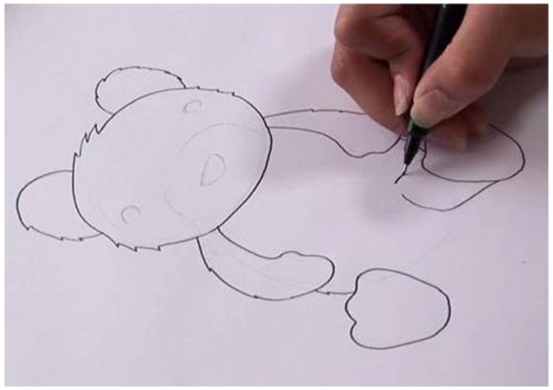
- Layout
- Formats
- Authentication
- Security
- Efficiency



jclouds

# Designing a Cloud 2.0 API

- Statefulness
- Progress & Errors
- Degradation
- Extensibility
- Evolution



## Consuming a Cloud 2.0 API

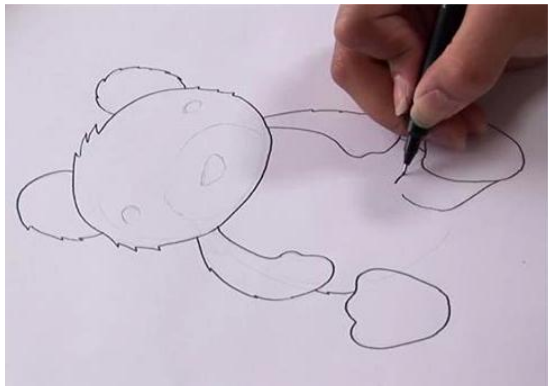
- Blocking vs. non-blocking
- Error handling
- Consistency
- Backoff

jclouds

## Designing a Cloud 2.0 API: Layout

✓ REST or some RESTish variant
  - "pure" REST: more self-documenting but more reference parsing

✓ context in scheme
  - watch for land-grab

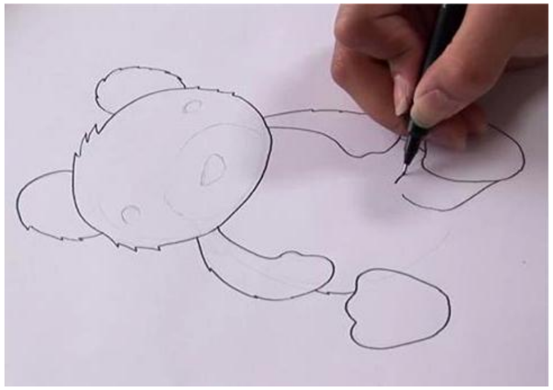


# Designing a Cloud 2.0 API: Formats

✓JSON
- Depends on library support

✓Query params
- Good for caching & auto-retry
- Easily logged

✓YAML
- Concise



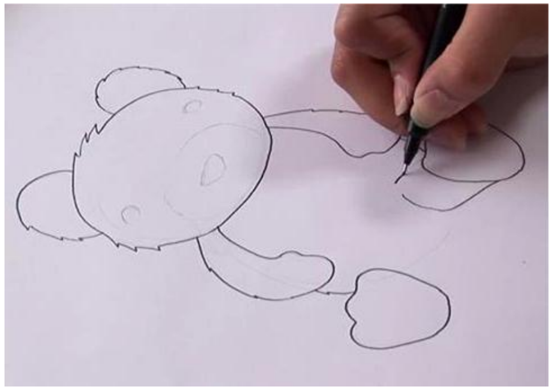
@jclouds

jclouds

## Designing a Cloud 2.0 API: Auth

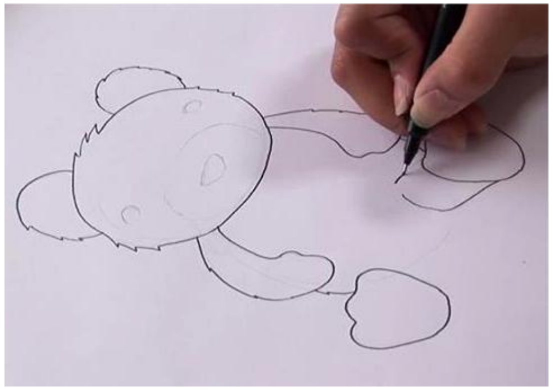
✓ API key & secret
- Can be rotated and safely lost
- De facto standard

✓ OAuth 2
- Weight behind it but better be easier than 1!



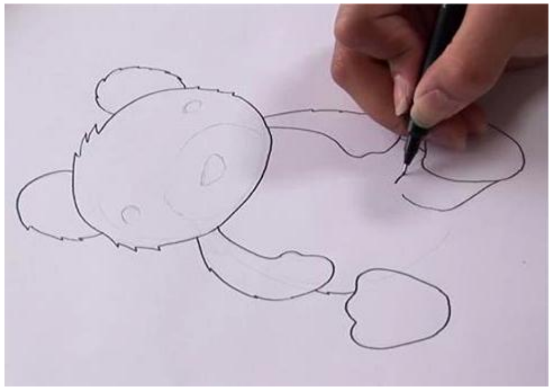
## Designing a Cloud 2.0 API: Security

- ✓ Offer HTTPS
  - Allow user to make the tradeoff
- ✓ Pre-signed requests
  - Enable side-loading
- ✓ Time windows



## Designing a Cloud 2.0 API: Efficiency

- ✓ Multi-part & ranges
  - Parallelization of large payloads
- ✓ Async calls
  - Return a job for everything
- ✓ Not just binary
  - “Can’t give you exactly what you requested but have this”

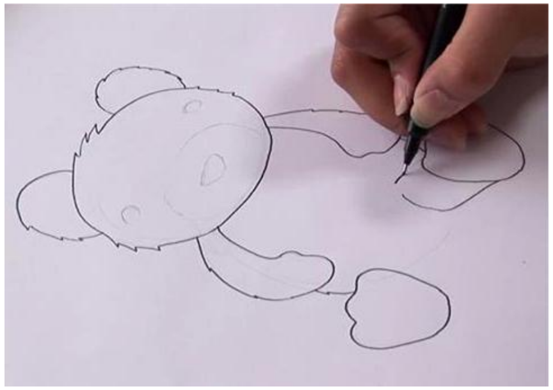


jclouds

# Designing a Cloud 2.0 API: Statefulness

✓ Idempotency

- Cloud is a volatile place
- “Default to action”

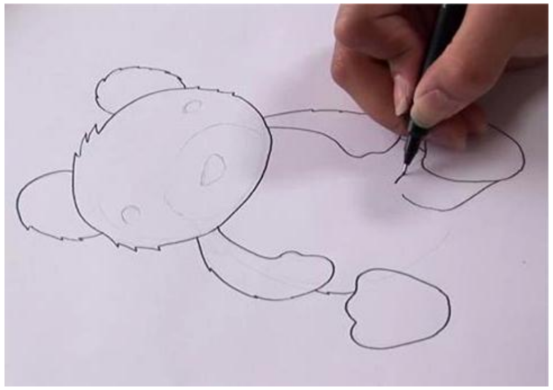


## Designing a Cloud 2.0 API: Progress

✓ Status requests

✓ Support cancellation

- Great for troubleshooting and saving $$$
- Application-level graceful degradation

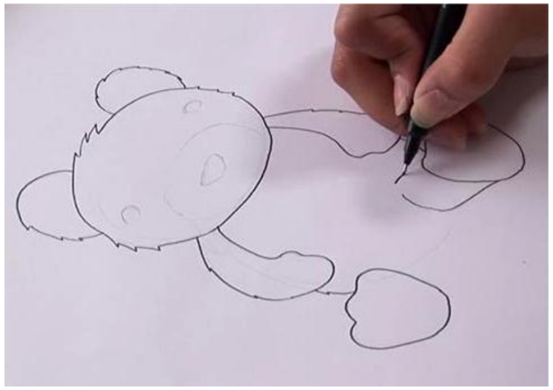


@jclouds

jclouds

# Designing a Cloud 2.0 API: Errors

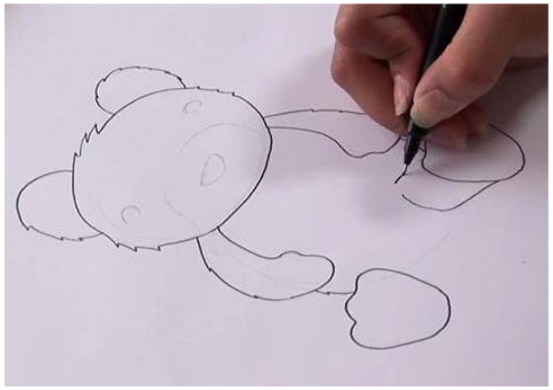
- ✓ Well-defined status codes
  - But must also be able to distinguish subcategories
- ✓ Error payload
  - E.g. backoff period or missing funds



jclouds

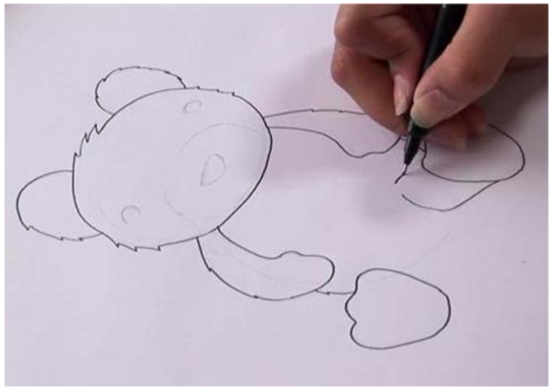
## Designing a Cloud 2.0 API: Degradation

- ✓ Throttling: Don't!
  - …if you can avoid it. Scale is the name of the game!
- ✓ Backoff request
  - Ask first
- ✓ Effective bandwidth controls
  - Implementation challenge



# Designing a Cloud 2.0 API: Extensibility

- ✓ Well-known sets
  - Ad hoc hard to consume

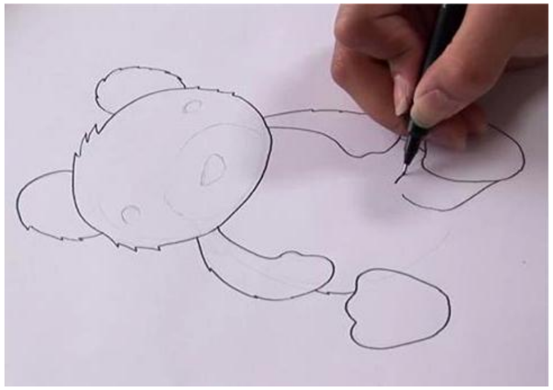


@jclouds

jclouds

# Designing a Cloud 2.0 API: Evolution

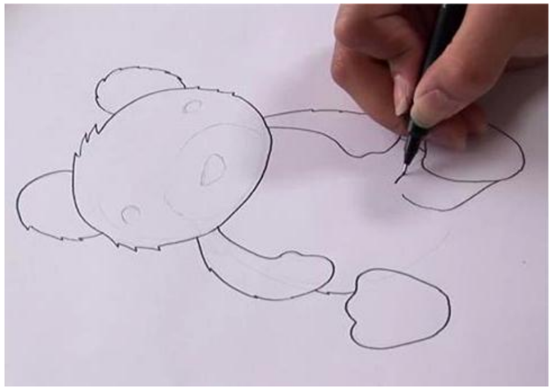
- ✓ Allow version checks
- ✓ Support different client versions
- ✓ Pre-release announcements
- ✓ Documentation
  - Examples ready to cut'n'curl
- ✓ Stability!



# Designing a Cloud 2.0 API: Evolution

✓Take Beta Seriously™
- Seek out tough clients
- Open up access
- Write integration drivers

✓ Friendly API licence
- E.g. Creative Commons



@jclouds

jclouds

## Consuming a Cloud 2.0 API: Blocking vs. non-blocking

✓ Block client-side
  - …if you need to. More control over cancellation etc.

✓ Beware throttling effects
  - …from async production or even test code!

@jclouds

jclouds

## Consuming a Cloud 2.0 API: Error handling

✓ Retry policies

- Be prepared for timeouts, spurious errors etc.
- Don’t want every exception handled in your application logic

@jclouds

jclouds

# Consuming a Cloud 2.0 API: Consistency

✓ Prepare for stale data!

- Metadata and content might be out of sync
- Need async and parallel testing to flush this out

jclouds

## Consuming a Cloud 2.0 API: Backoff

✓ Prioritize operations

- Prepare for degraded performance
- “Low cash” mode

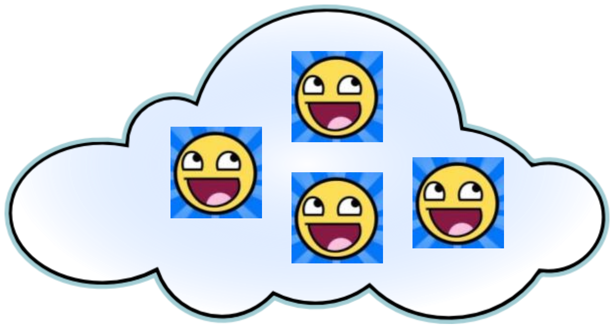
jclouds

# More Info

- http://www.jclouds.org
- http://github.com/jclouds/jclouds

jclouds

## Come Join In!



- jclouds@googlegroups.com
- @jclouds
- IRC #jclouds on freenode