1. Put your Webapp in the Cloud
2. Virtual Observatory
3. Web Services

**django in the cloud**
- Google App Engine
  - Run on Google machines
  - Your own domain or.appspot.com
  - Java or Python
  - Persistent storage
  - Free but many quotas
  - Scalable – advantage of the cloud
  - Does one thing well: running web apps
- Amazon
  - LAMP based
  - not so easy with django ...

**App Engine Does One Thing Well**
- App Engine handles HTTP(S) requests, nothing else
  - Think RPC: request in, processing, response out
  - Works well for the web and AJAX; also for other services
- App configuration is dead simple
  - No performance tuning needed
- Everything is built to scale
  - “infinite” number of apps, requests/sec, storage capacity

**Security**
- Prevent the bad guys from breaking (into) your app
- Constrain direct OS functionality
  - no processes, threads, dynamic library loading
  - no system (use webtech API)
  - can’t write file (use datastore)
  - disallow unsafe Python extensions (e.g., ctypes)
- Limit resource usage
  - Limit 1000 files per app, each at most 1MB
  - Hard time limit of 50 seconds per request
  - Most requests must use less than 300 msec CPU time
  - Hard limit of 1MB on request/response size, API call size, etc.
  - Quota system for number of requests, API calls, emails sent, etc.
  - Everything an app does is limited by quotas; for example:
    - request count, bandwidth used, CPU usage, datastore call count, disk space used, emails sent, even errors!
    - If you run out of quota that particular operation is blocked (raising an exception) for a while (~10 min) until replenished

**Why Not LAMP?**
- Linux, Apache, MySQL/PostgreSQL, Python/Perl/PHP/Ruby
- LAMP is the industry standard
- But management is a hassle:
  - Configuration, tuning
  - Backup and recovery, disk space management
  - Hardware failures, system crashes
  - Software updates, security patches
  - Log rotation, cron jobs, and much more
  - Redesign needed once your database exceeds one box
- "We carry pagers so you don’t have to"
placeholdtext
step 4: register, upload, run

Virtual Observatory

Toward a “new astronomy”
• Past:
  Observations of small, carefully selected samples

Toward a “new astronomy”
• Multi-wavelength data for millions of objects
• Data Mining, Outliers, Correlations, etc
• Theory vs Experiment

Toward a “new astronomy”
• Tools
• Visualization
• Data Mining
Virtual Observatory is:

- Uniform interfaces
  - standards
- Data discovery
  - & publication
- Data federation
  - science is in the join
- Usability
  - "I just want to ...."
- Remote analysis
  - big data, services, storage
- Big computing
  - terabytes & teraflops

OpenSkyQuery

Web Enabled Source Identification
with Cross-Matching (WESIX)

Upload images to SExtractor and cross-correlate the objects found with selected survey catalogs.

This NVO service does source extraction and cross-matching for any astrometric FITS image. The user uploads a FITS image, and the remote service runs the SExtractor software for source extraction. The resulting catalog can be cross-matched with any of several major surveys, and the results returned as a VOTable. The web page also allows use of Aladin or VOPlot to visualize results.

Example Service

Web Services

- Principle: Click or Code
- Protocol: simple REST/GET or sophisticated SOAP
- Standards
  - Basic service profile
    - formal description (WSDL), input parameters, output formats, capabilities
    - VOResource (who and what is it)
  - Security (single sign-on, authentication and authorisation)
  - Asynchronous (Batch) services
  - Distributed data storage (VOSpace)
- Coordinated with IVOA, Open Grid Forum

Service Oriented Architecture
Service Oriented Architecture

VO Data Services
- Cone Search
  - First standard NVO service:
    - radius=position ⇒ list of objects
    - encoded as VOTable
- Simple Image Access Protocol
  - "cone search for images"
  - images are referenced by URL
- Simple Spectrum Access Protocol
  - spectra have subtleties ⇒ protocol more complicated

VO Data Services
- Astronomical Data Query Language
  - For database queries
  - Core SQL functions plus astronomy-specific extensions
    - Sky region, Xmatch
- SkyNode
  - Exposes relational databases
  - Accepts ADQL query
  - "Full" SkyNodes support positional cross-match function
  - OpenSkyQuery portal
    - show database structure
    - query tools

Registry
- publish – find – bind
- Registry Metadata
  - Descriptions of data collections, data delivery services, organizations, etc.
  - Based on Dublin Core with astronomy-specific extensions
  - Represented as XML schema; extensible
  - Contents stored in Resource Registries
    - exchange metadata records through the Open Archives Initiative Protocol (OAI-PMH)

Registry Content:
Datasets, Standard Services, Organizations

Identifier format
http://ivo://ivo.openarchives.org/describe?dataset=2MASS
Distributed Registry

Web Service Glossary

- Web service (think a (code) library on the web)
  - A piece of software available over a network with a formal description of how it is called and what it returns that a computer can understand
- REST (think HTTP GET and POST)
  - An approach to web services that uses the existing infrastructure of the web
- SOAP (think envelopes)
  - An approach to web services that uses an XML-based messaging framework
- WSDL (think a contract)
  - An XML description of a web service (normally SOAP) and how to interact with it

Web services in the VO

- SkyNodes, Open SkyQuery and WESIX
- Footprint Services
- Spectrum Services
- Registry Interface
- VOSpace
- CasJobs
- CDS
- VO Services
- Cone search

What we won’t cover

- Security:
  - I want to protect my data/resources
- Attachments
  - I want to upload/download a file
- State
  - I want the service to remember things
- Asynchrony
  - It’s going to take some time
- Messaging
  - I want more than just request/response

Numquam ponenda est pluralitas sine necessitate

- REST (Representation State Transfer) is an architectural style not an implementation:
  - Distinguishes resources and operations
  - Each resource is identified by a URI
  - The only operations are the HTTP methods

REST by analogy

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>Think</th>
<th>Description</th>
<th>/proc analogy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT</td>
<td>CREATE</td>
<td>Create a resource with the user specified id</td>
<td>Start a new process</td>
</tr>
<tr>
<td>GET</td>
<td>RETRIEVE</td>
<td>Retrieve a resource representation</td>
<td>Get the status of a process</td>
</tr>
<tr>
<td>POST</td>
<td>UPDATE</td>
<td>Update a resource/Append to a resource/Create a resource with a server assigned id</td>
<td>Amend a process</td>
</tr>
<tr>
<td>DELETE</td>
<td>DELETE</td>
<td>Delete a resource</td>
<td>Kill a process</td>
</tr>
</tbody>
</table>
By accident, not by design

- Parameters in HTTP GET URLs:
  - `http://processes.com/123/status`
- HTTP GET is safe – it does nothing else than retrieval
- HTTP GET, PUT and DELETE are idempotent – the effects on the system of one or N identical requests are the same
- Consider HTTP GET with:
- Services which maintain idempotency whilst allowing parameterized URLs are accidentally RESTful
- 85% of web service traffic, 6x faster allegedly

When to use REST

- RESTful services are good for WWW-type functionality
- Limitations:
  - No formal description:
    - WSDL not widely used yet
  - Not an issue for pure REST (no parameters in URL)
  - No messaging infrastructure:
    - Reliable messaging
    - Message routing
  - No message level security:
    - Digital signatures
  - No resource lifecycle management
  - No transaction support
  - No asynchronous event notification

What is SOAP?

- Simple Object/Service-Oriented Access Protocol
- W3C specification
  - `http://www.w3.org/TR/soap`
- An XML-based messaging framework for exchanging information between peers in a decentralized, distributed environment
- Defines the message structure but not the message content (needs other technologies)
- Fundamentally stateless (no memory of what has happened previously)
- One-way message exchange paradigm

What is WSDL?

- Web Services Description Language
  - W3C specification
    - `http://www.w3.org/TR/wsdl`
- An XML grammar for describing the public interface of a web service in terms of:
  - its exposed operations
  - the message formats
  - protocol bindings (e.g. SOAP, HTTP)
- Most commonly used with SOAP
  - defines format of message content in the SOAP body
**Anatomy of a WSDL**

A WSDL document describes:

- a set of services which exchange messages
- the data in the messages is defined in a set of types
- each message transfer in a service defines an operation
- operations have access to specific implementations using protocols like HTTP

**What is WSDL**

It is the shared document that allows a request-response conversation

---

**Using SOAP and WSDL**

- Start with WSDL *(contract-first)* vs. start with code *(contract-last)*
- Frameworks provide APIs:
  - to handle SOAP:
    - De/serialization: converting code object to/from XML representation
  - and code bindings derived from WSDL:
    - Stub: client-side proxy to service
    - Skeleton: server-side to handle request and response
- Alternate approach:
  - SOAP (and WSDL) are just XML
  - Use XSLT, curl, JavaScript, etc.