GOVERN YOUR CLOUDS

Mist.io is an open platform for managing heterogeneous infrastructures

GET STARTED

One UI to rule them all: Manage your Docker containers along with your servers and VMs

3 September 2014

Docker has been making headlines lately when they announced Docker 1.0. Soon after, Google released their Kubernetes project, dubbed "an open source implementation of container cluster management" targeting large scale cluster management based on Docker. Suddenly, all the big players started jumping in. IBM, Red Hat, CoreOS and, strangely, even Microsoft.

The truth is, Docker is pretty awesome. While everyone else's focus was on the Cloud and virtualization, Docker came out with a different approach: containment. Leveraging the power of the Linux kernel for resource isolation (cgroups) and LinuX Containers (LXC), Docker provides image management and deployment services for applications.

Docker's power stems from the fact that it can package an application and its dependencies in a virtual container that can run on any Linux server on top of the Docker engine. That container can be easily transferred and run anywhere, whether on premise, public cloud, private cloud, bare metal, etc. Compared to a VM, a Docker container is much smaller and thus easier to transfer and deploy.

And while virtualization targets infrastructure scalability, Docker is ideal for quickly upscaling or downscaling your application, according to your needs. Eventually, VMs and Docker containers aim at solving different parts of the equation and end up playing side by side.

Here in Mist.io, we have already been using Docker in our Continuous Integration suite to help test and deploy our code faster and so far we've been loving it. Using Jenkins, Ansible and Docker we are able to test new code faster and in more environments, which has greatly minimized the risk to introduce bugs and regressions when deploying new features.

One drawback of adding Docker containers to our toolchain was that we now had one more layer to manage along with our mix of bare metal servers and cloud VMs. And while there are already some powerful open-source Docker management tools like Shipyard, UI fragmentation can lead to overhead and eventually frustration. So we decided to take action to help ourselves and give something back to the open-source community.

Mist.io's free and open-source cloud management UI, already supported bare metal servers, public and private clouds. By adding Docker support we ended up with a central place for all our management needs. We can now use it to launch and destroy new containers, search for docker images in our private and the official registry and even set up monitoring and automation for our containers. To try it out, you can use our hosted service at https://mist.io or install it locally. Lets see how it works.

We'll assume you already have a Docker engine up and running. To set up the engine itself, you can follow Docker's installation guides.

To connect to Docker remotely, we'll need to configure the daemon to listen on a TCP port. By default the Docker daemon listens on unix:///var/run/docker.sock, so in order to be able to connect we have to start the docker server like this:

```
/usr/bin/docker -d -H tcp://0.0.0.0:4243 -H unix:///var/run/docker.sock
```

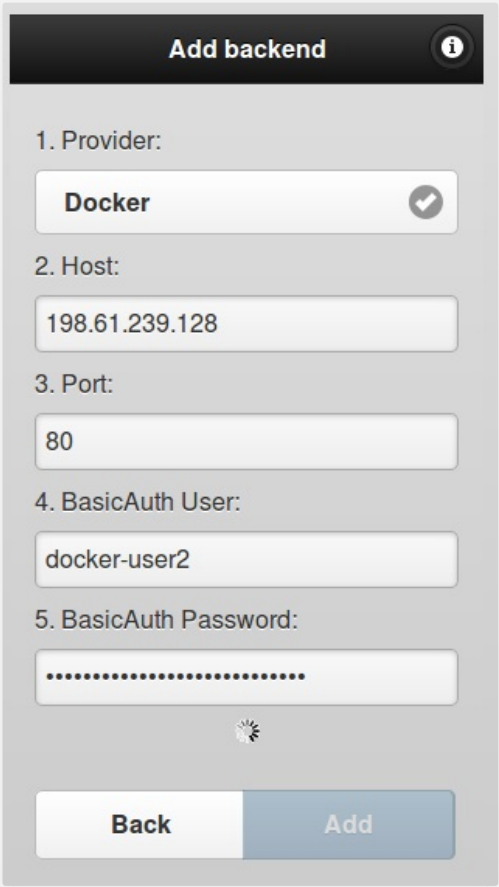
This leaves the server open to anyone that can connect to that port. Unfortunately there is not a default way to password protect the API calls. We will apply http basic auth, and password protect our docker api behind a web server by proxying the 4243 port (this step is optional, but highly recommended unless you are using other means of protection). First, lets set up a password file by running:

```
htpasswd -c /etc/nginx/.htpasswd docker-user2
```

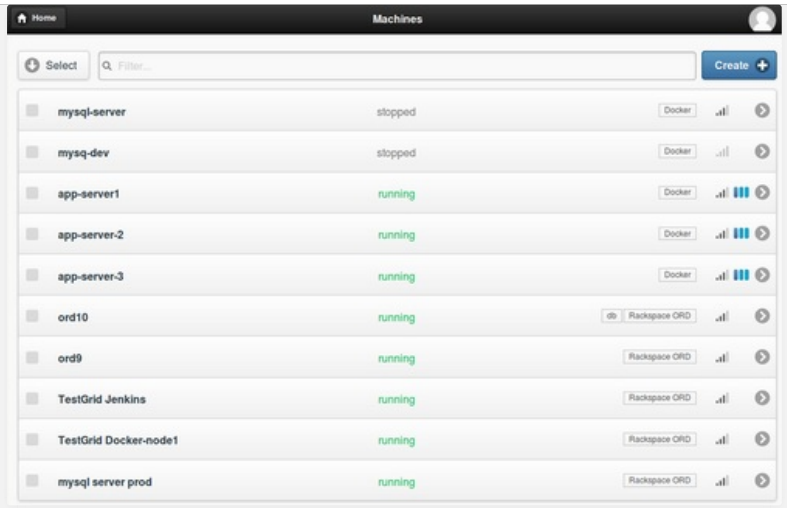
and typing the password. Next, we'll enable auth to our nginx configuration by adding:

```
proxy_pass http://127.0.0.1:4243;
auth_basic "Restricted";
auth_basic_user_file /etc/nginx/.htpasswd;
```

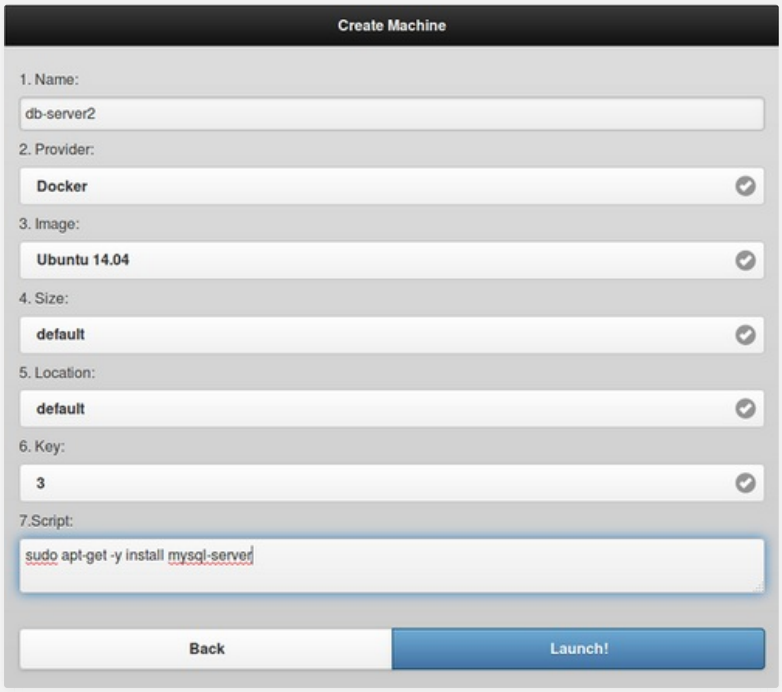
Once we've set that up, all we'll add our Docker backend to Mist.io:



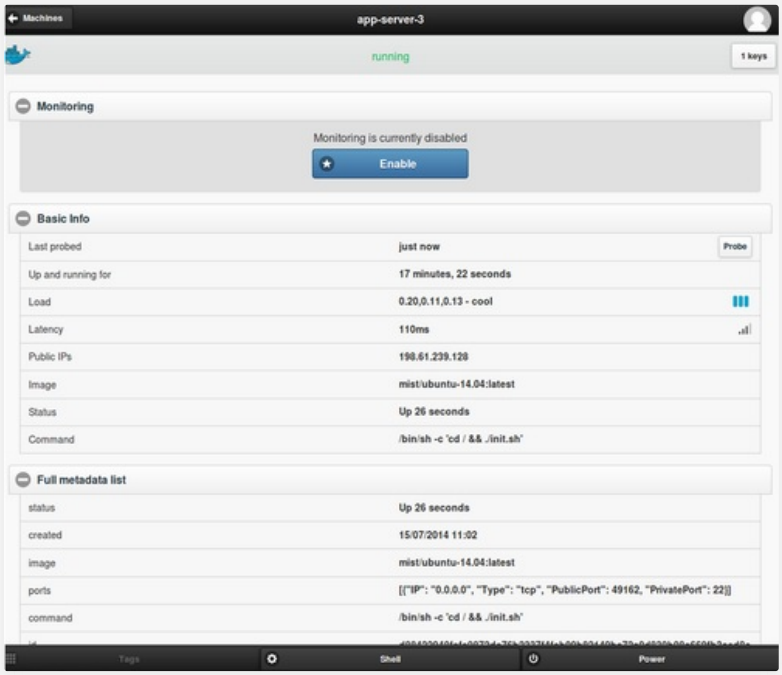
All we need is the IP address and port of the Docker engine, and optionally the username and password that we protected it using http auth. Once we add our Docker backend, all the containers will be added to our machines list, neatly tagged as Docker containers to be easily distinguished from the rest of our infrastructure.



Creating a new container is a few seconds away. Click on the Create button and fill out the form. If the image you select allows it, you can specify an SSH key to be deployed on the container and a post-deploy script to be run.

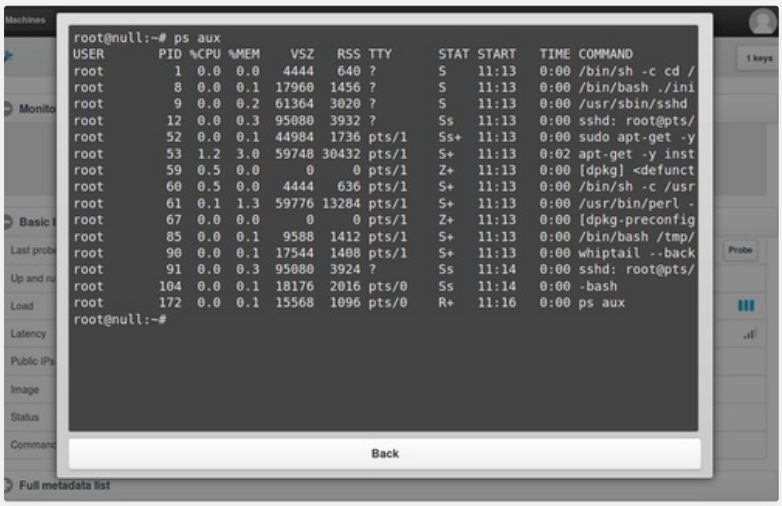


Click on the Launch button and the new container will appear in the list of machines managed by Mist.io.

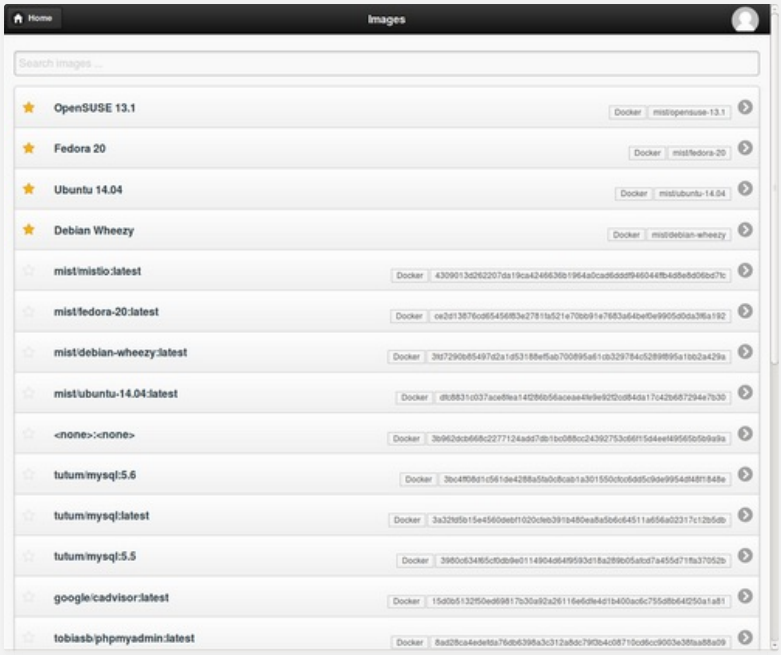


If our container was deployed with an SSH key, the touch-friendly web shell will be

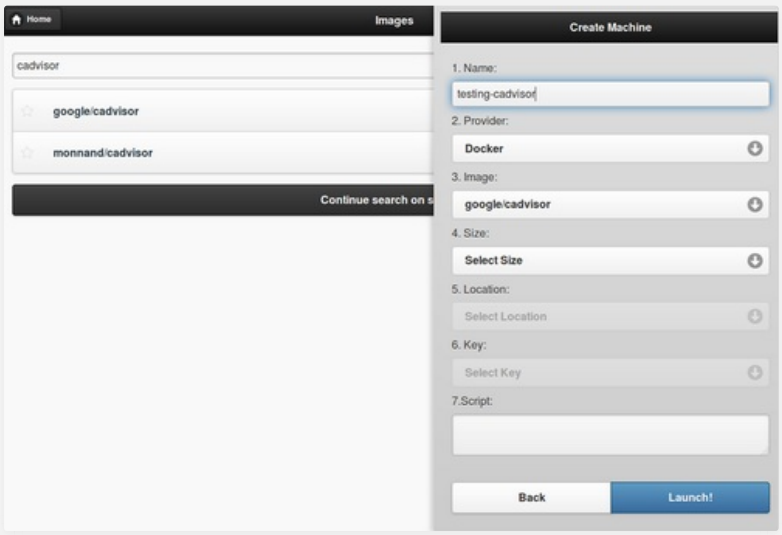
available



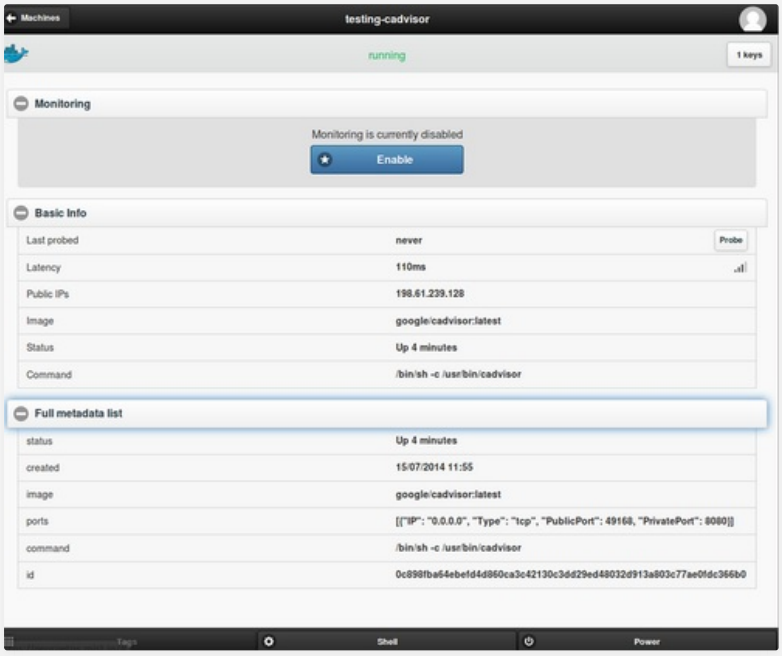
One of Docker's most important features that can make our life much easier, is the ability to use images to bootstrap our containers. With Mist.io, you get a central place to find and use Docker images. Clicking on the Images link, will give you a listing that includes the default Mist.io images, along with the images that are available on your Docker engine.



More importantly, searching for images will return results not only from the local Docker images, but also from the official Docker registry. For example, launching Google's cadvisor container is as simple as searching for "cadvisor", clicking on one of the images and adding a name for it.



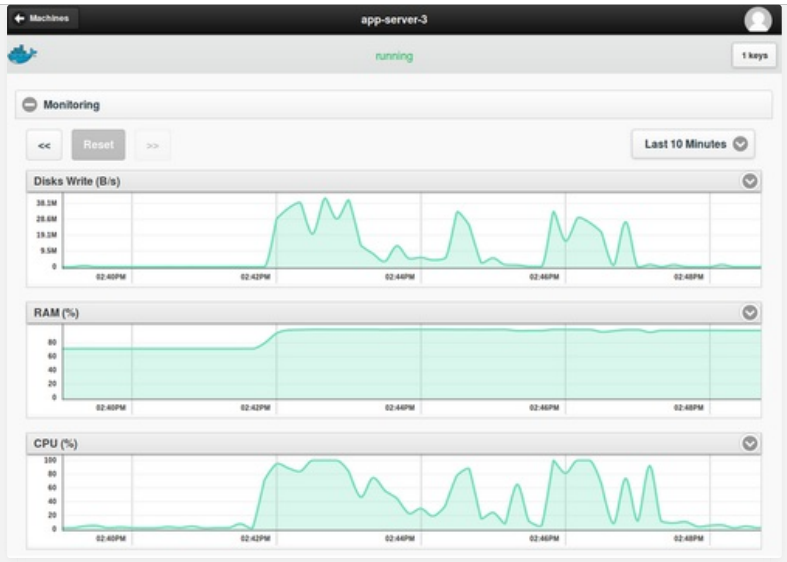
When you click on the Launch button, Mist.io will tell Docker to pull the image from the registry, create the container and start running it. Let's check out if it worked. Click on the newly created container to see more info.



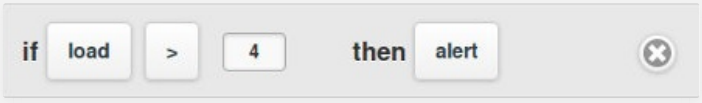
The container's public IP is 198.61.239.128 and on the Full metadata list, we can see that the public port cadvisor is configured to listen to, is 49168. If we visit that address, we should see cAdvisor up and running.

Enabling monitoring and automation for your docker containers is just a couple of clicks away. The UI can connect to our monitoring service and all you need is an account at https://mist.io.

To enable monitoring and alerting, just click on the Enable monitoring button. If the container is accessible though SSH, Mist.io will automatically install and configure the open-source collectd monitoring agent. Otherwise, it will prompt you with instructions on how to install it manually. Once the agent is up and running, we'll start getting data into our graphs.



Being able to see how your container was performing in the past week is great. But It's even more important to get notified right away once things start getting out of the accepted norms. Click on the Add Rule button just below the graph to set up your first alerting rule. E.g. if you want to get an email once the load exceeds 4, your rule should look like this:



If you suspect that you have a process that's leaking memory, you can set up Mist.io to restart that process once your memory usage exceeds 95%.

Thats it. You've successfully set up monitoring for your container and added your first alerting rules. You can continue to add more monitoring metrics and alerts or write your own custom ones.

In the future, we plan to extend Docker support to allow the removal of images and import/build images from Dockerfiles and URLs. If you have any ideas or feature requests, drop us an email at support@mist.io or clone the github repo and start hacking away!

TRY MIST.IO NOW

- #Docker
- #cloud computing
- #monitoring
- #open source
- #opensource
- 1 year ago
- 1
- 1 Comment
- Permalink

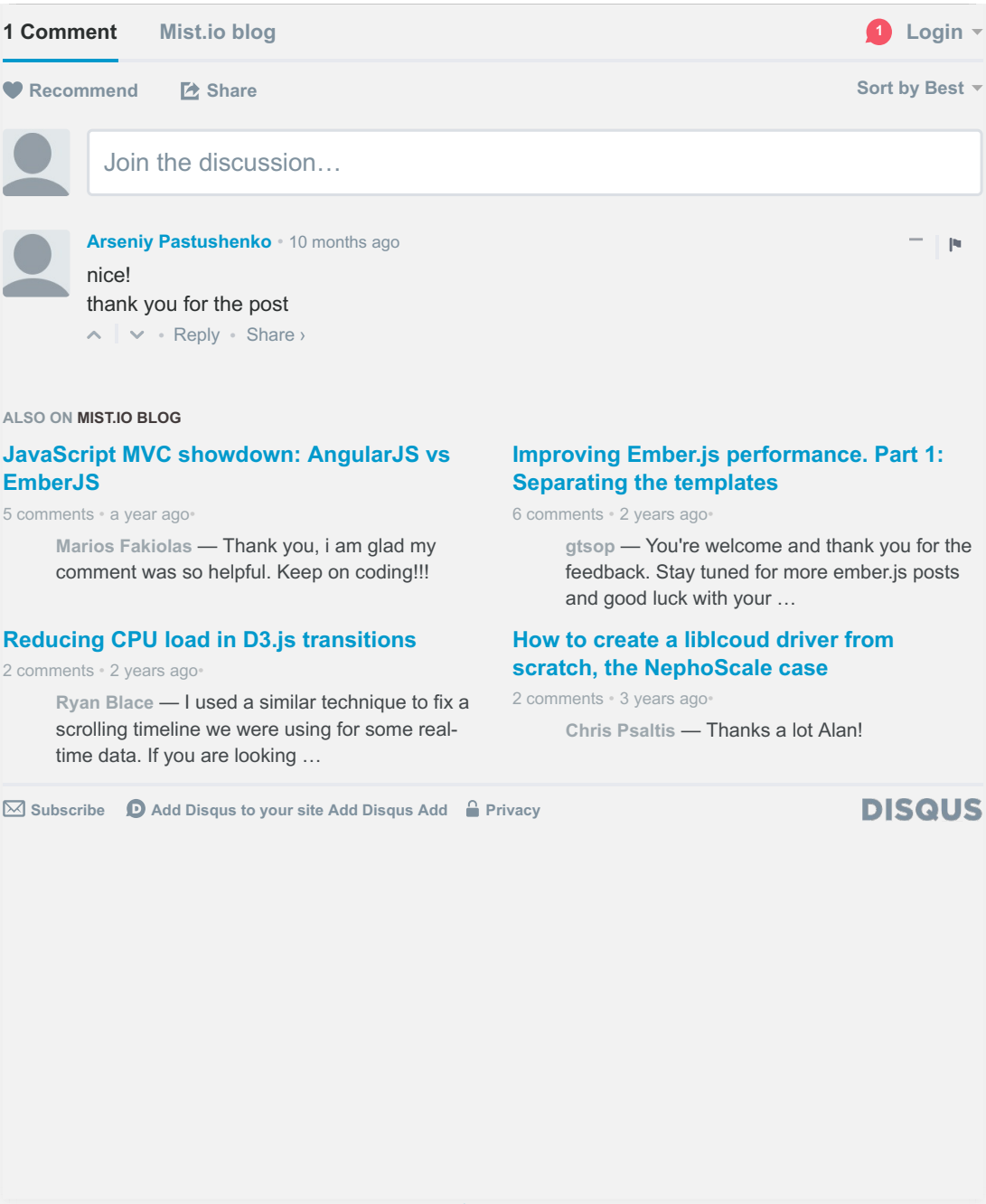
1 Notes/ Show

Recent comments

Share

Short URL
https://tmblr.co/Zaozys1P

TwitterFacebookPinterestGoogle+



← Previous • Next →

Search

Post Navigation

« Previous Post · Next Post »

Newsletter

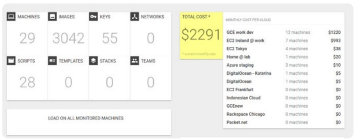
Stay up to date, get new posts in your email.

email address SUBSCRIBE

Tweets by @mist_io

mist.io @mist_io

Real time cost reporting for your infrastructure: bit.ly/29Fd6f2



14 Jul

mist.io @mist_io

All the awesome presentations from #OpenStackIL at bit.ly/1swYbgD And here are some photos if you missed it bit.ly/1UFry9y

RSS Random Archive Mobile