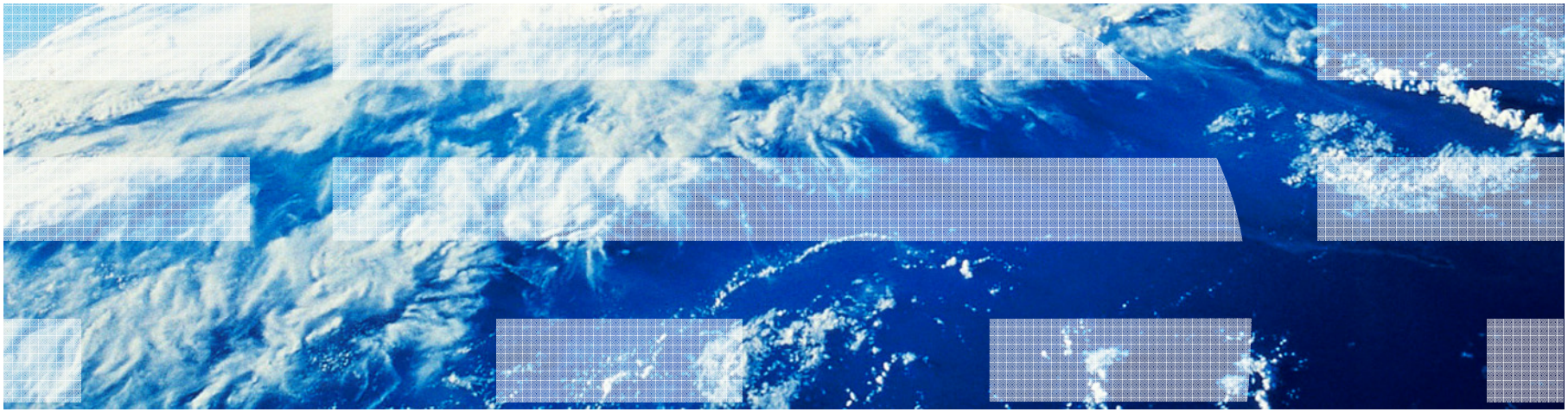
Storlet Engine for Executing Biomedical Processes within the Storage System

Simona Rabinovici-Cohen, Ealan Henis, John Marberg, Kenneth Nagin

{simona, ealan, marberg, nagin}@il.ibm.com

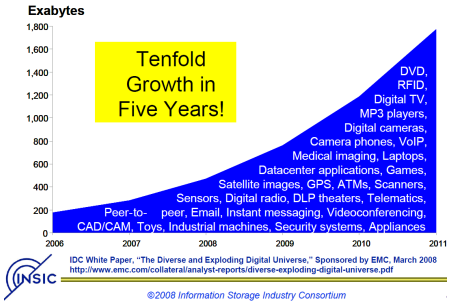


7th International Workshop on Process-oriented Information Systems in Healthcare (ProHealth'14)

Emerging Trends Motivate Storlets

- **"Data is the new Oil"**
  - In its raw form, oil has little value
  - Once processed and refined, it helps power the world

- **Data deluge of biomedical data**
  - Medical images, genomic sequencers, videos, etc.
  - Multiple large objects created by various devices



- **Object storage for content depots generally:**
  - Utilizes large bandwidth to serve big data over the WAN
  - Uses commoditized of-the-shelf hardware with under utilized CPUs

→ **Process and refine the data where it is stored**

  - Create a computational object storage with **storlets**

Storlets

- The concept of storlets was first introduced in our MSST 2007 paper to offload data-intensive computations to the long term archival storage

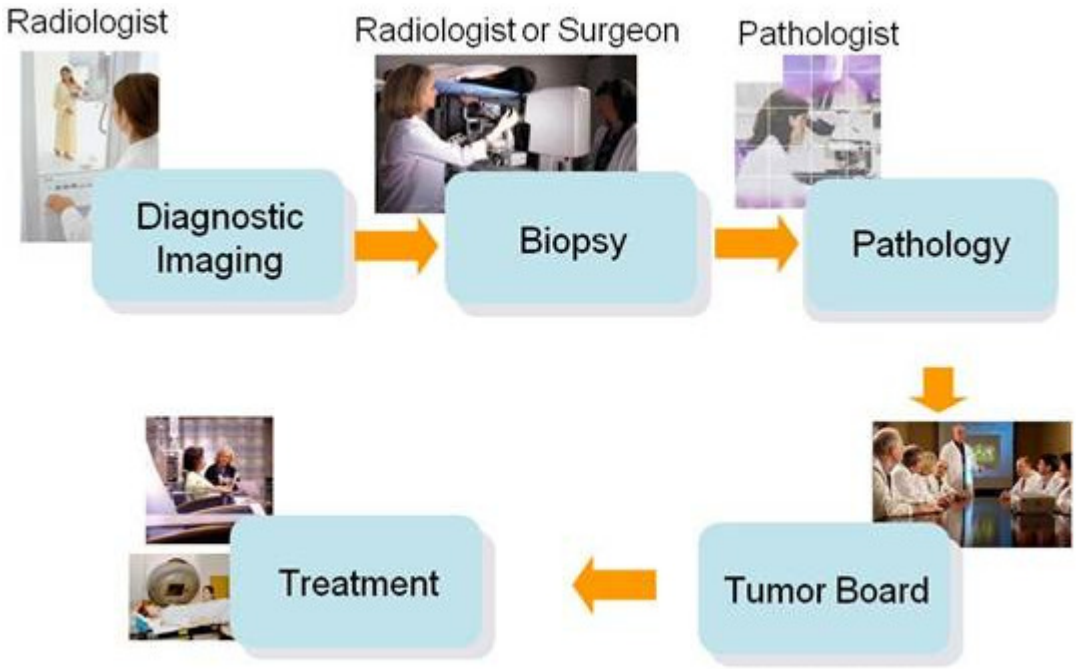
What is a storlet?

- Dynamically uploadable computational module running within the storage close to the data
- Executed in a sandbox when not trusted
- Analog to stored procedures in databases

The Storlet Engine mechanism:

- Adds flexibility to the storage
- Makes the storage extensible - makes the storage a platform
- Transforms the storage from “keeping data” to “producing value from the data”

Pathology Department Use Case

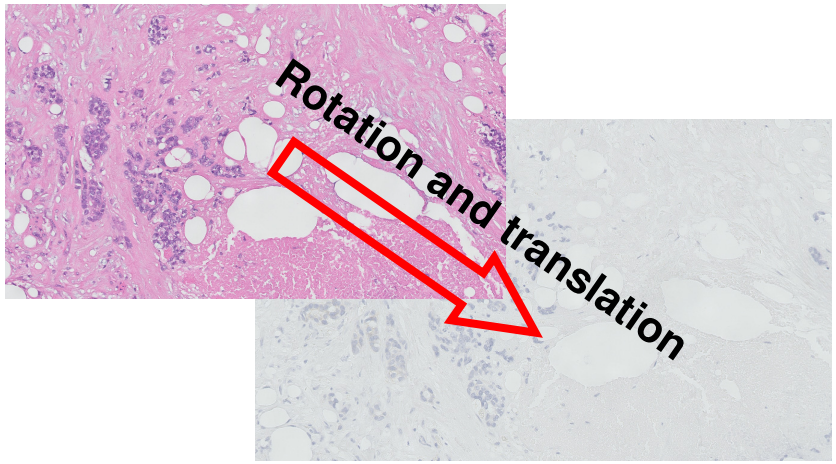


* image courtesy of Philips

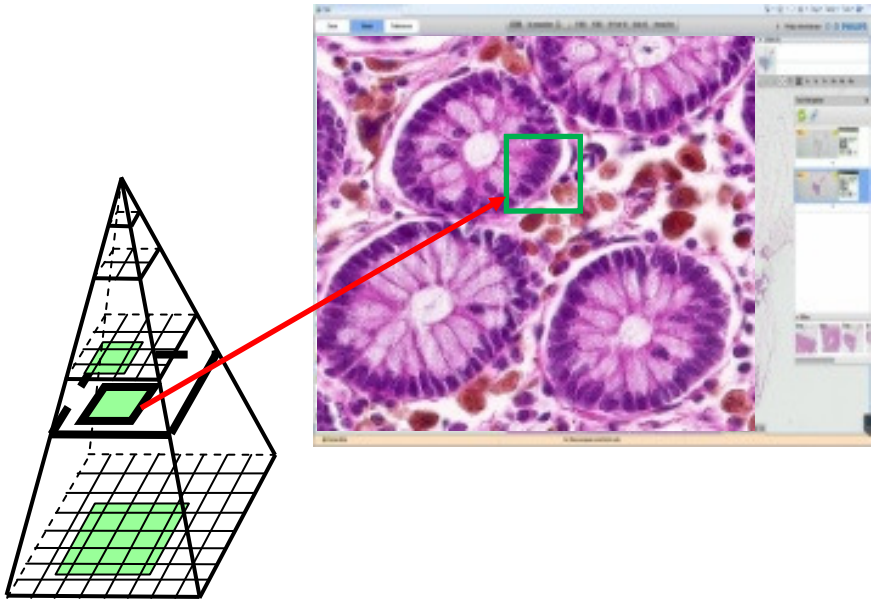
- A **tissue block** is given to the Pathologist
- Very thin **slices** can be cut off the tissue block and mounted on glass slides
- The thin slices are stained with different stain, e.g. HER2, PR, ER, etc.
- Pathology **images** are created for the stained slices
- Large images can be 200K x 200K which can consume 5-10 GB

ROI Extraction Business Process

Aligning images via rotation and translation

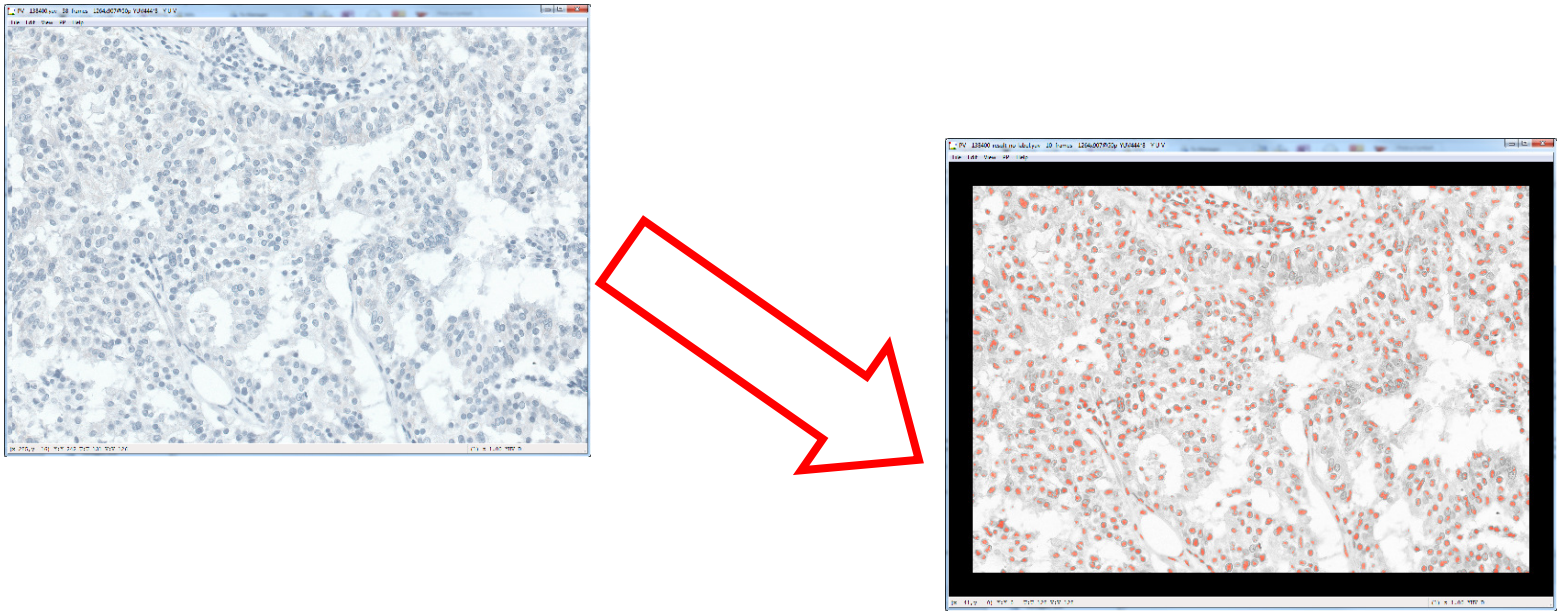


Extracting subset of the images and transforming to a standard format



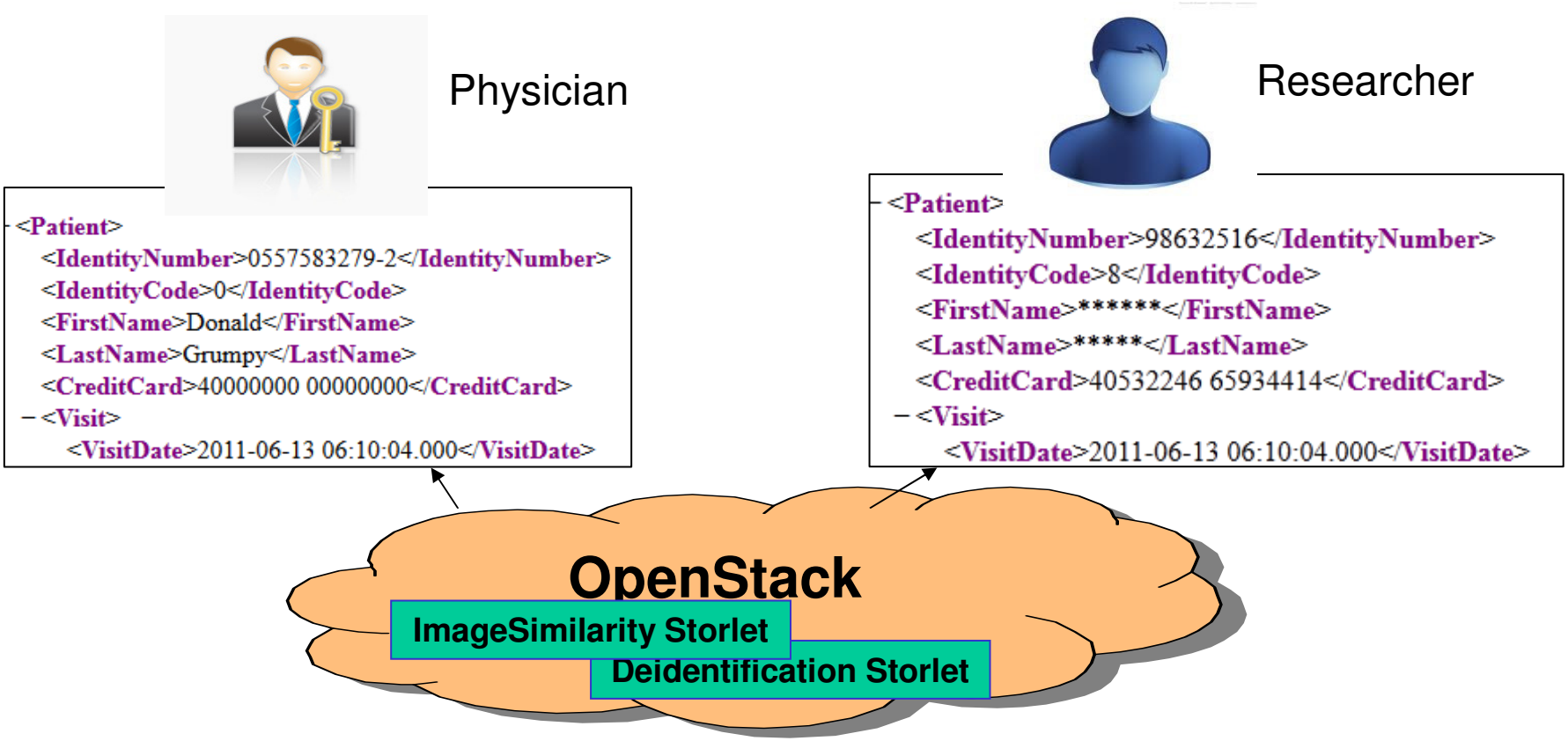
* images courtesy of Philips

Cell Detection Business Process



- Cell detection requires pattern search/feature extraction and analysis of pathology images
- This is very important for Pathologists because it provides a quick overview of the disease cells, based on which diagnosis can be made
- The oncologist machine may be too weak to execute this module
- With storlets technology, the Cell Detection Storlet will run in the cluster of machines within the storage

Cohort Identification Business Process



- A researcher would like to get a cohort of records and images with similar features from as multiple patients as possible
- This is more secure than the traditional method as it spares the need to move clear data to the staging system
- Helps comply with HIPAA

Client Value for Using Storlets

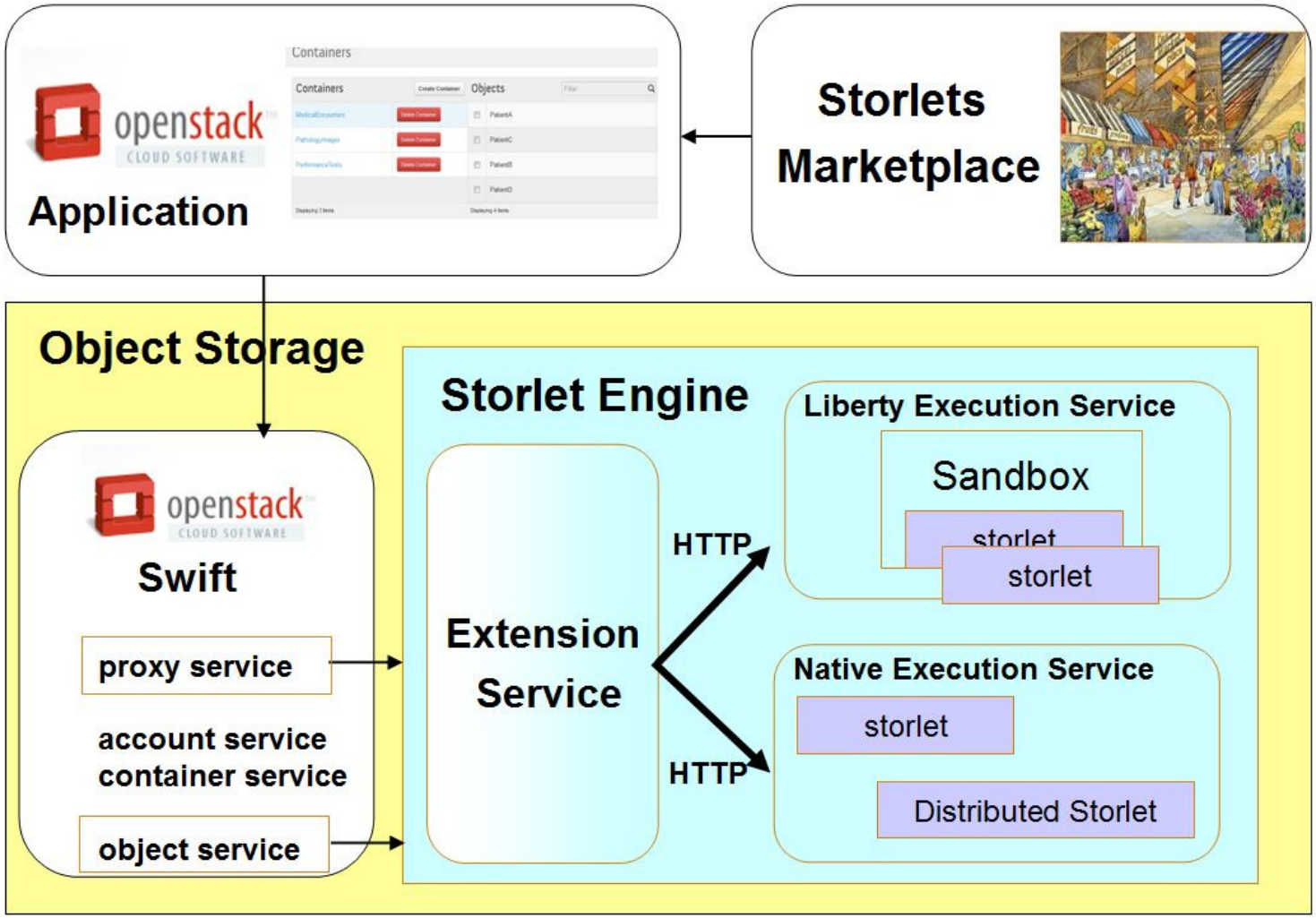
- **Reduce bandwidth** – reduce the number of bytes transferred over the WAN
  e.g. Analytics storlet

- **Enhance security** – reduce exposure of sensitive data
  e.g. De-identification storlet

- **Save costs** – consolidate generic functions that can be used by many applications while saving infrastructure at the client side
  e.g. Curation storlet

- **Support compliance** – monitor and document the changes to the objects and improve provenance tracking
  e.g. Transformation storlet

Storlet Engine Architecture



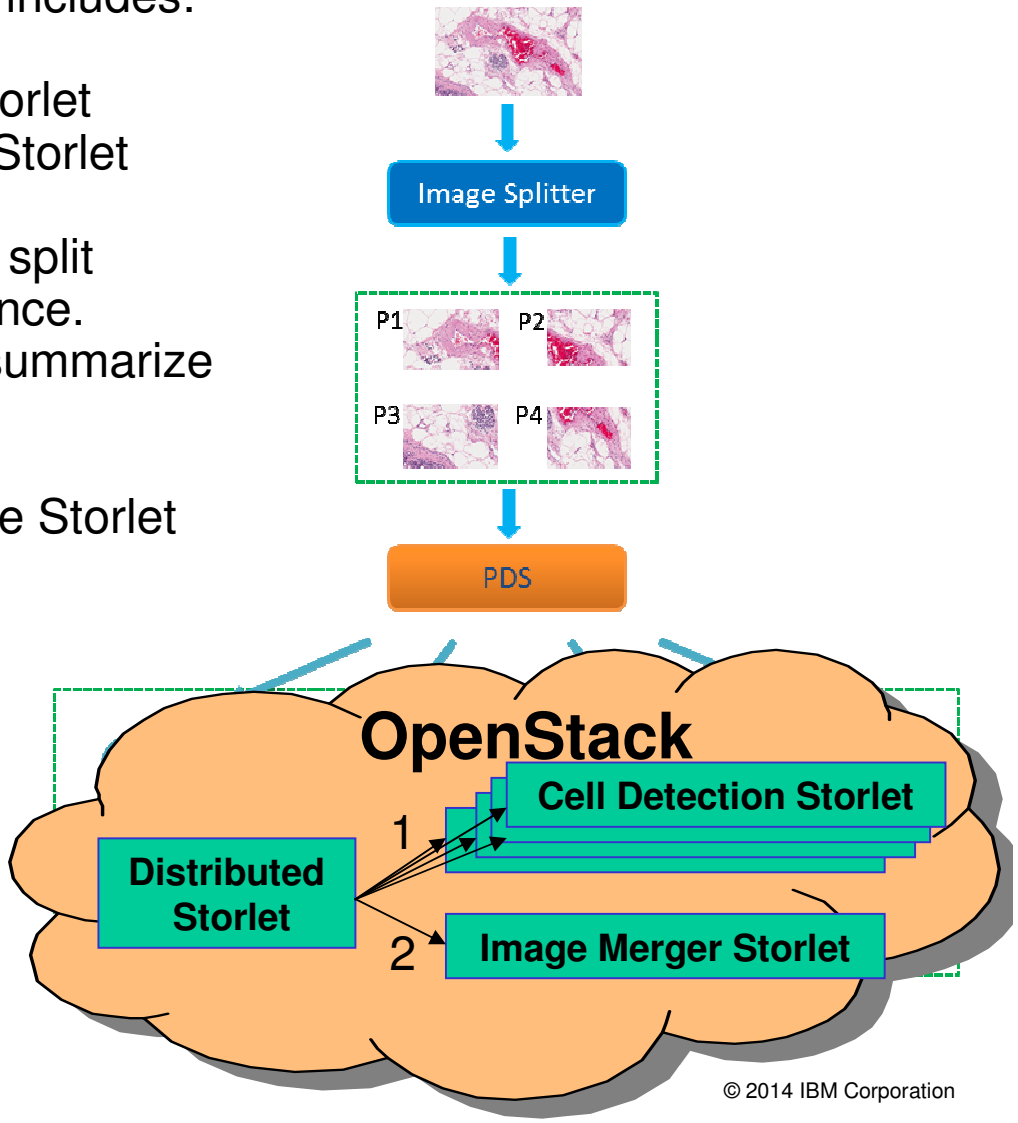
* The Native Execution Service can execute docker-based storlets

Distributed Storlet and Cell Detection

- The compound distributed storlet input includes:
  - list of data objects to work on
  - split storlet e.g. CellsDetectonStorlet
  - merge storlet e.g. ImageMergerStorlet

- The distributed storlet initiates multiple split storlets on each data object at its residence. Afterwards, it calls the merge storlet to summarize the results.

- The distributed storlet is a service in the Storlet Engine suitable for analytics processes.



- **Distributed Storlet and Image Merger Storlet run in Proxy node**
- **The multiple Cell Detection Storlets run in Object nodes**

Rules Mechanism

- Enables automatic conditional invocation of storlets
  - Explicit storlet activation overrides implicit activation
  - Rules kept as per tenant editable object, with specified access control
  - Configured by tenant, user, role, container, object, content_type
  - Wildcards (“*”) allowed in a rule (high flexibility)
  - The first rule that matches the input is activated – prioritized list of rules

- Examples:
  - De-Identification (per Role)
  - Transformation (per Content Type)
  - Fixity (per Container)

Security Model

Requirements

- Restrict and control the execution capabilities of storlets (per sandbox type)
- Isolate and protect storlets of different tenants from each other (multi-tenancy)
- Authorize and verify access from storlets to object storage

Solution Features

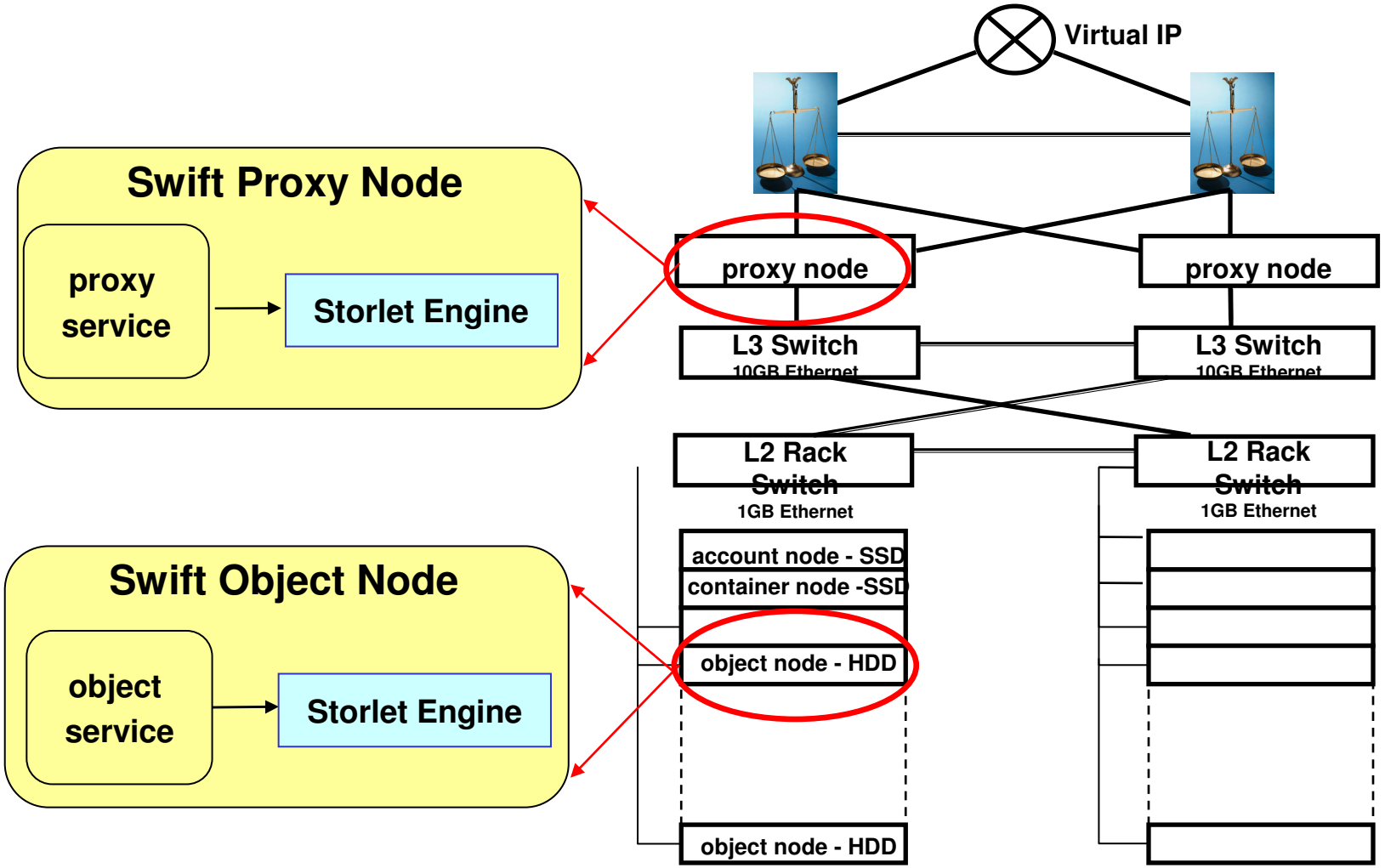
- Use a separate web server for each combination of [tenant, sandbox]
- Dispatch each storlet request to the server of the given tenant and sandbox
- Leverage Linux capabilities for protection and isolation of individual web servers
  - Each server is associated with dedicated unique Linux uid/gid and port number
  - Server and all its storlets run in a single process, deployed under the unique uid/gid
  - Server listens on its unique port number
  - Sensitive files have the server's unique uid/gid permissions
  - Communication to target ports/hosts is filtered by iptables using uid and gid rules
- Storlets use tokens to access the object storage (supplied as request parameters)

Sandbox Types

- A storlet may be associated with different trust levels:
  - Storlet can be written by a system administrator; written by a user; or bought as part of a third-party package or downloaded from some site.
  - Storlet execution can be initiated at different privilege levels: by an administrator; by a privileged user; or by a regular user.
  - Based on the source of the storlet, the initiator, and the storlet functionality, a certain level of trust should be associated with the storlet.

- Storlets can run in various types of sandboxes to associate different levels of trust with different storlets:
  - **Admin Sandbox** – the storlet can do all operations.
  - **User Sandbox** - the storlet is restricted and cannot perform operations such as write to the filesystem, open a socket or access another host directly, spawn threads, etc.

Storlet Engine Deployment in OpenStack Swift

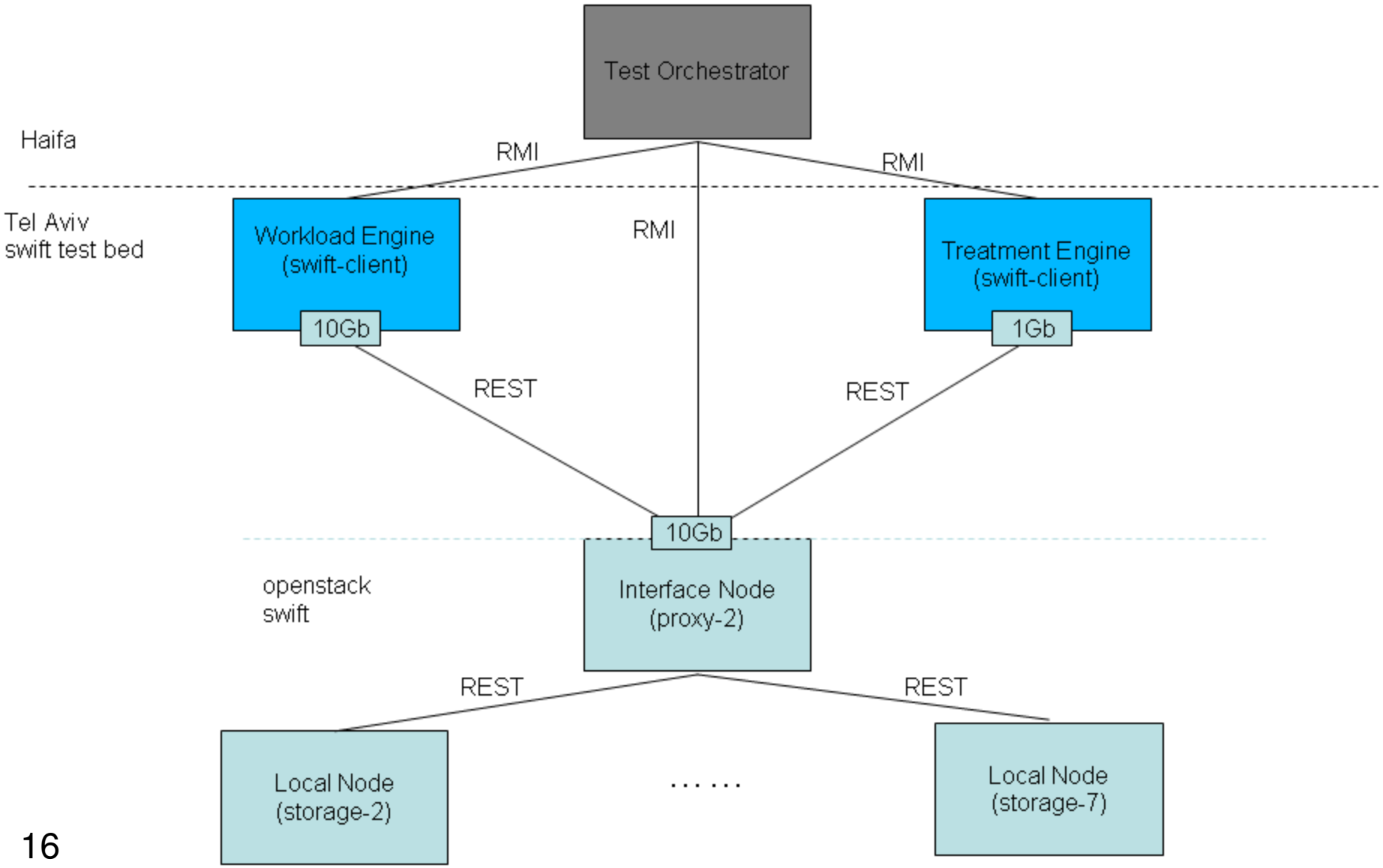


Performance Study: Goals

Answer these questions:

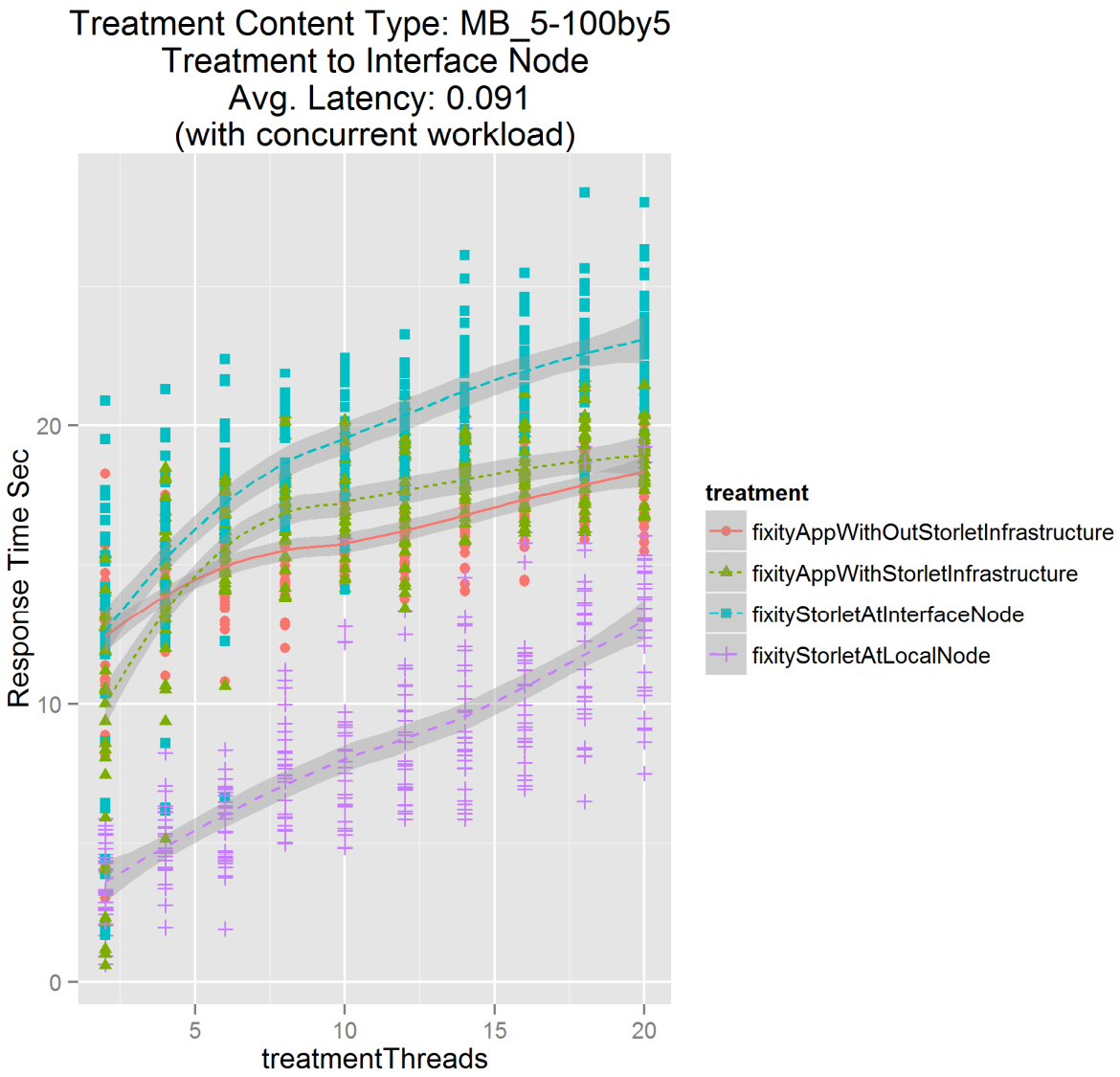
- What performance benefits can be derived by wrapping a function as a Storlet?
- How Storlets affect system performance?
- Performance for Storlets on local storage nodes versus Storlets on interface node?
- What host resources are most affected by Storlet?
- Do the performance issues change for private cloud (LAN) or public cloud (WAN)?

Test setup

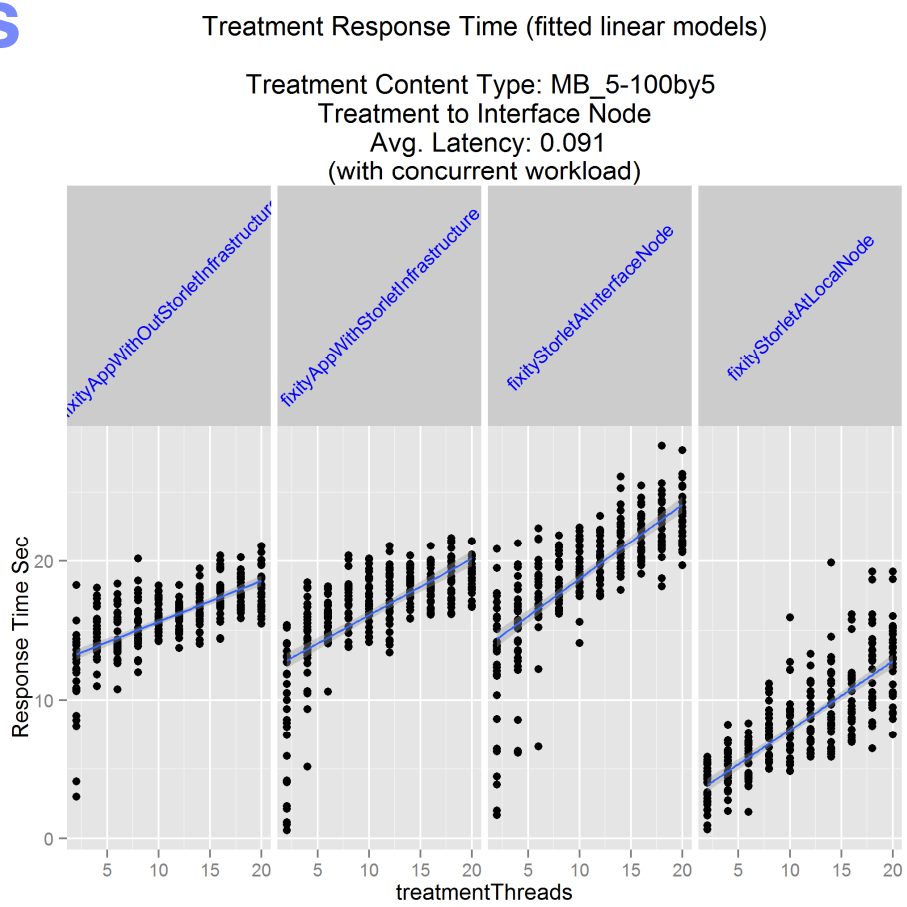

Test Orchestrator
Haifa
RMI
RMI
Tel Aviv
swift test bed
Workload Engine
(swift-client)
10Gb
RMI
Treatment Engine
(swift-client)
1Gb
REST
REST
10Gb
openstack
swift
Interface Node
(proxy-2)
REST
REST
Local Node
(storage-2)
. . . . .
Local Node
(storage-7)

Fixity Test Results

Treatment Response Time



Fixity Test Results



	Intercept	Slope	R.squared	std-error	correlation
fixityAppWithOutStorletInfrastructure	13	0.29	0.479	0.017733	0.692
fixityStorletAtInterfaceNode	13	0.54	0.557	0.02784	0.746
fixityStorletAtLocalNode	2.9	0.5	0.609	0.023062	0.781
fixityAppWithStorletInfrastructure	12	0.41	0.443	0.026465	0.666

Conclusions

- Introduced storlets for biomedical processes e.g.
  - ROI Extraction
  - Cell Detection
  - Cohort Identification

- Presented the Storlet Engine, an environment supporting computations within storage system
  - Architecture and key features of the system
  - Rules mechanism for automatic conditional invocation of storlets
  - Security model that supports storlets multi-tenancy and various types of sandboxes
  - Performance study of the Storlet Engine prototype

- Implemented a Storlet Engine prototype for OpenStack Object Storage (code-named Swift) in the context of the ENSURE and ForgetIT EU projects

Acknowledgement

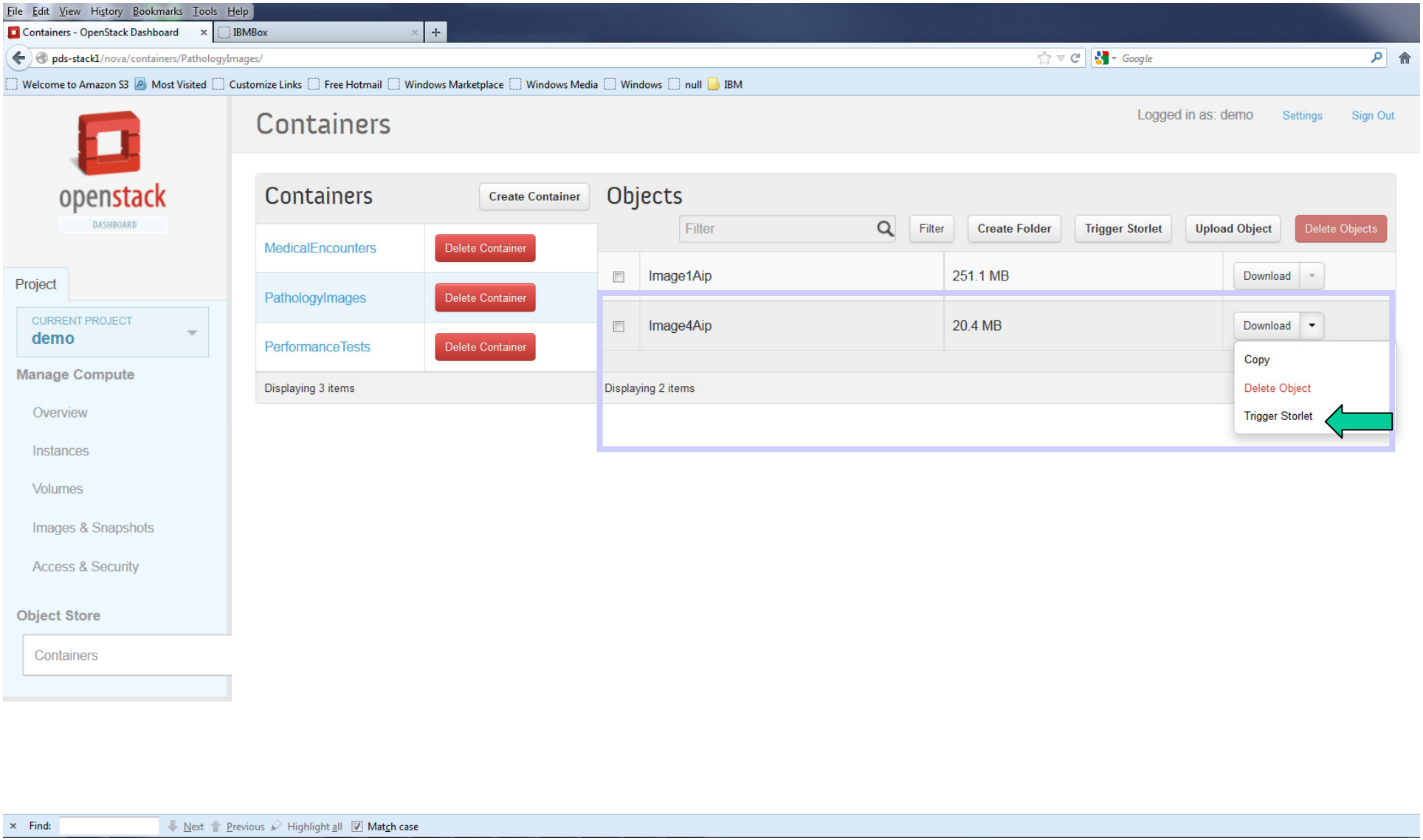
The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement 270000 and under grant agreement 600826.

http://www.ensure-fp7.eu/

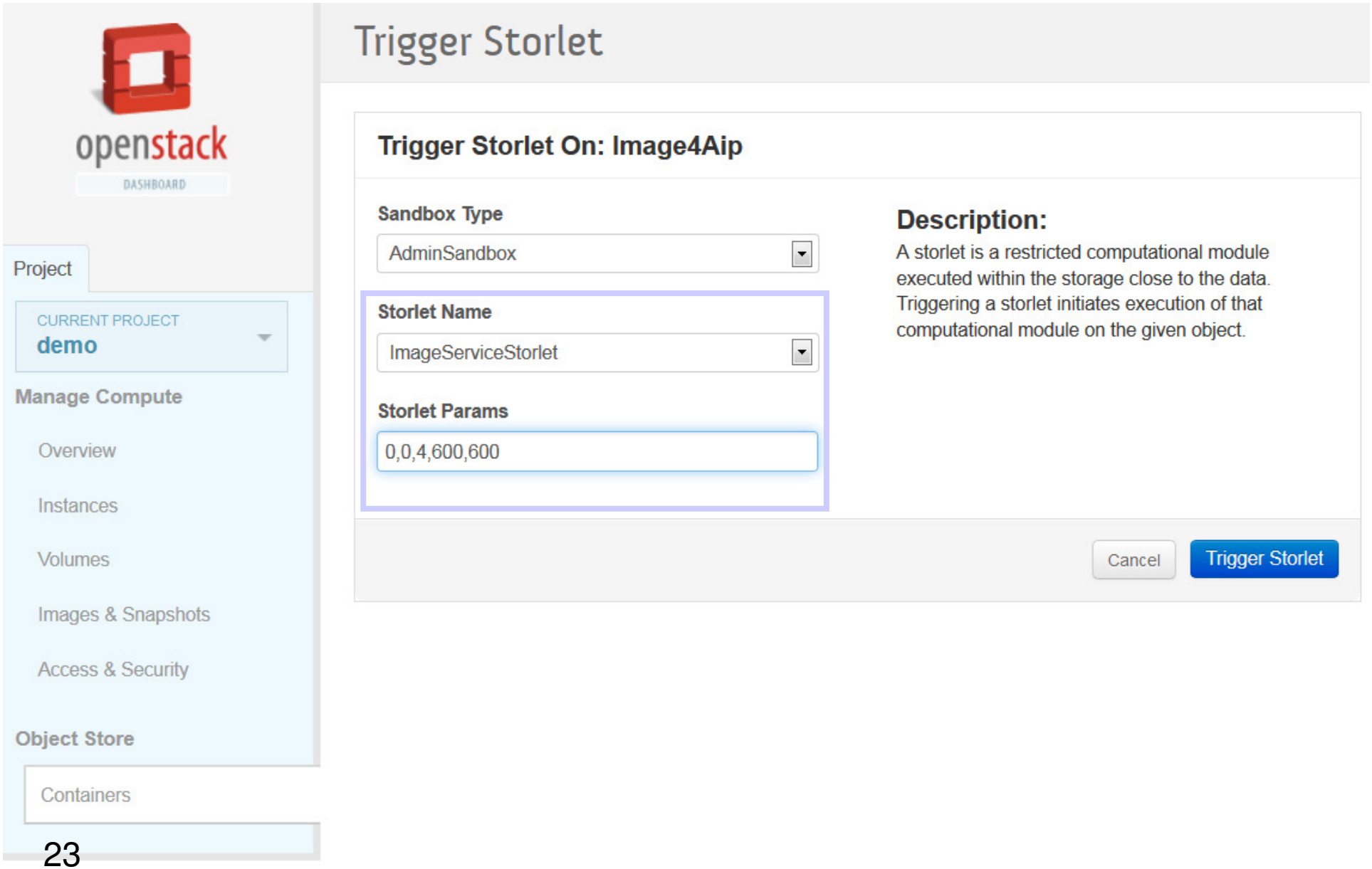
http://www.forgetit-project.eu/

DEMO

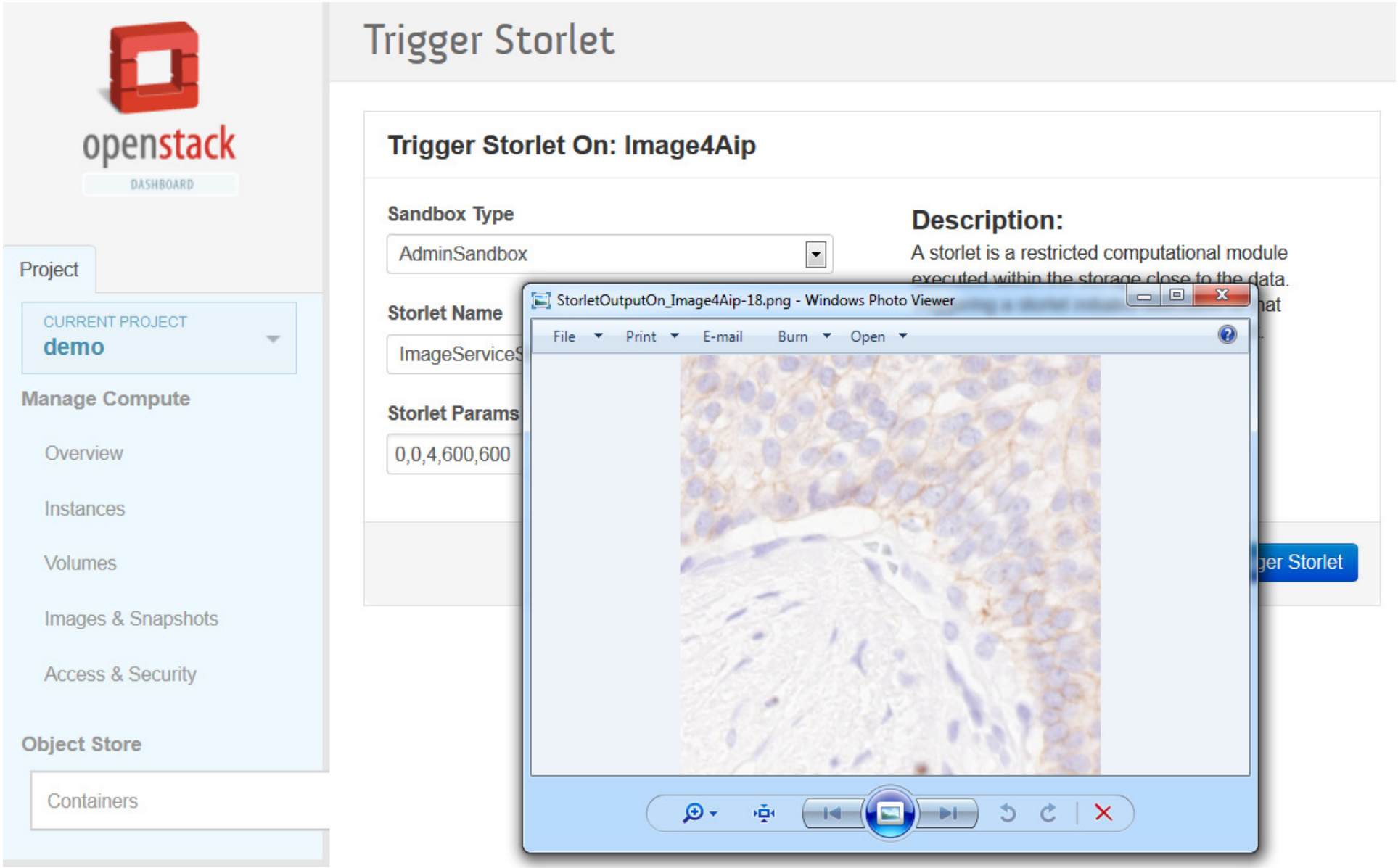
ROI Extraction: Trigger Storlet Selection



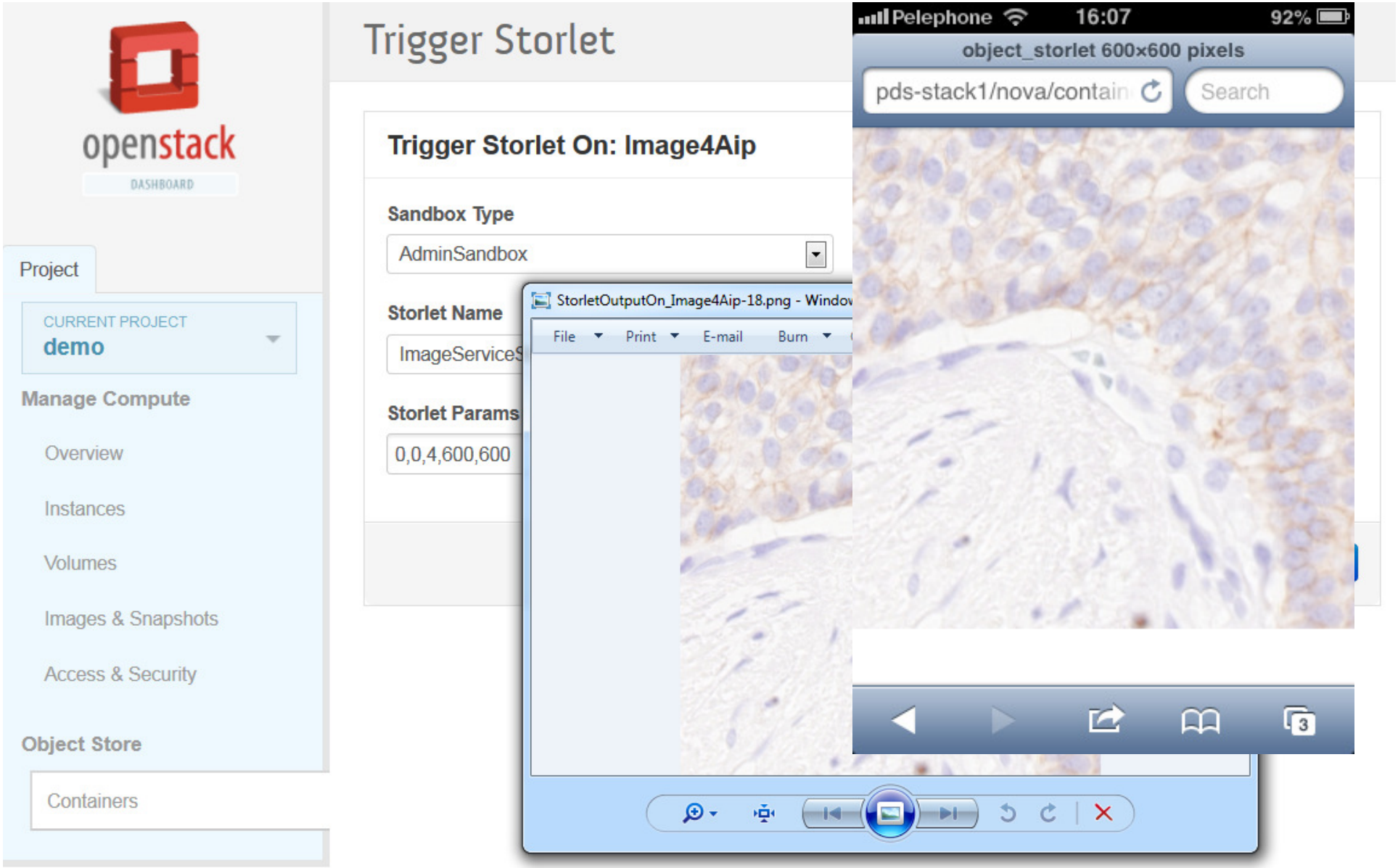
ROI Extraction: Trigger Storlet Window



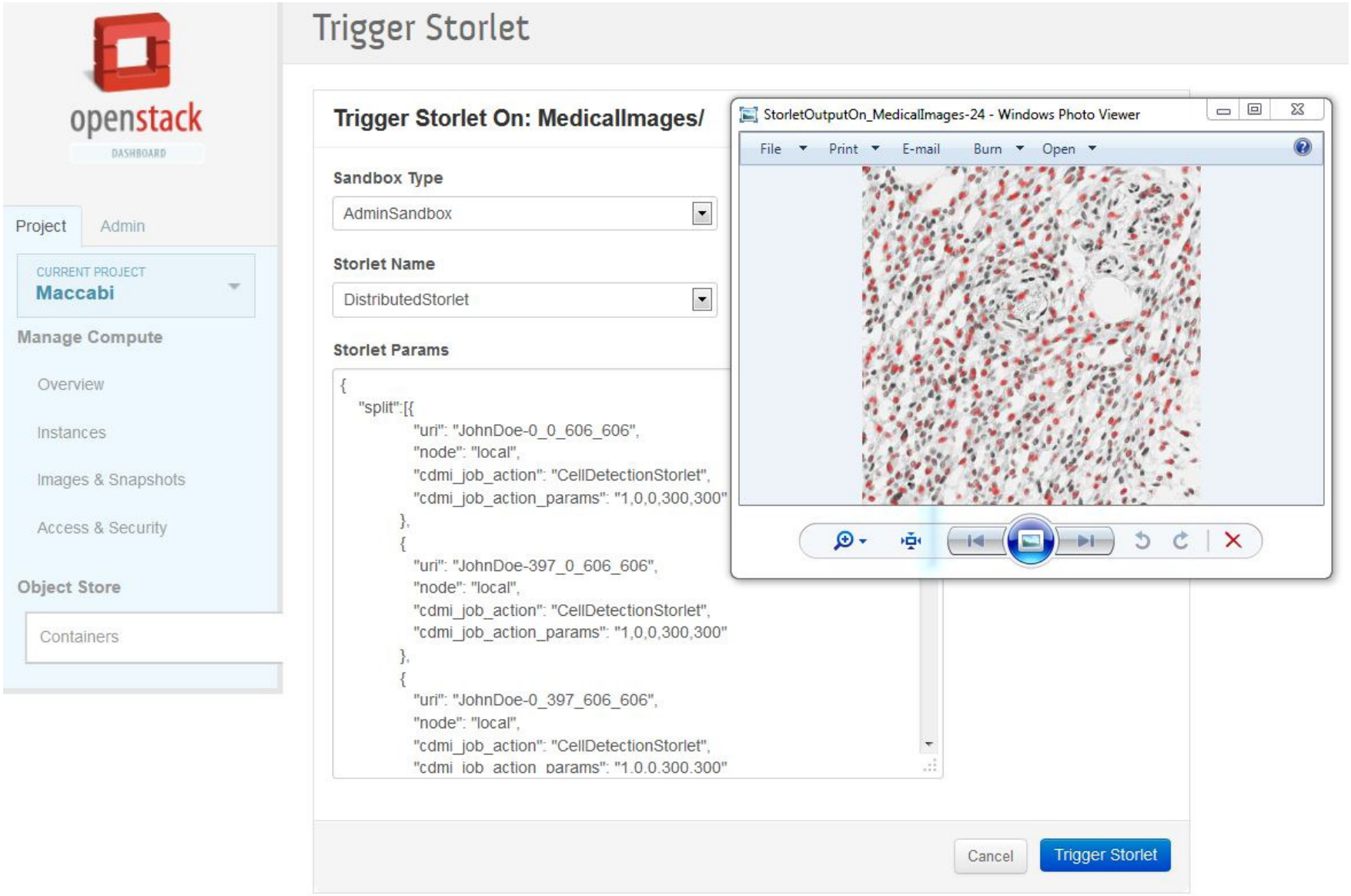
ROI Extraction: Trigger Storlet Result



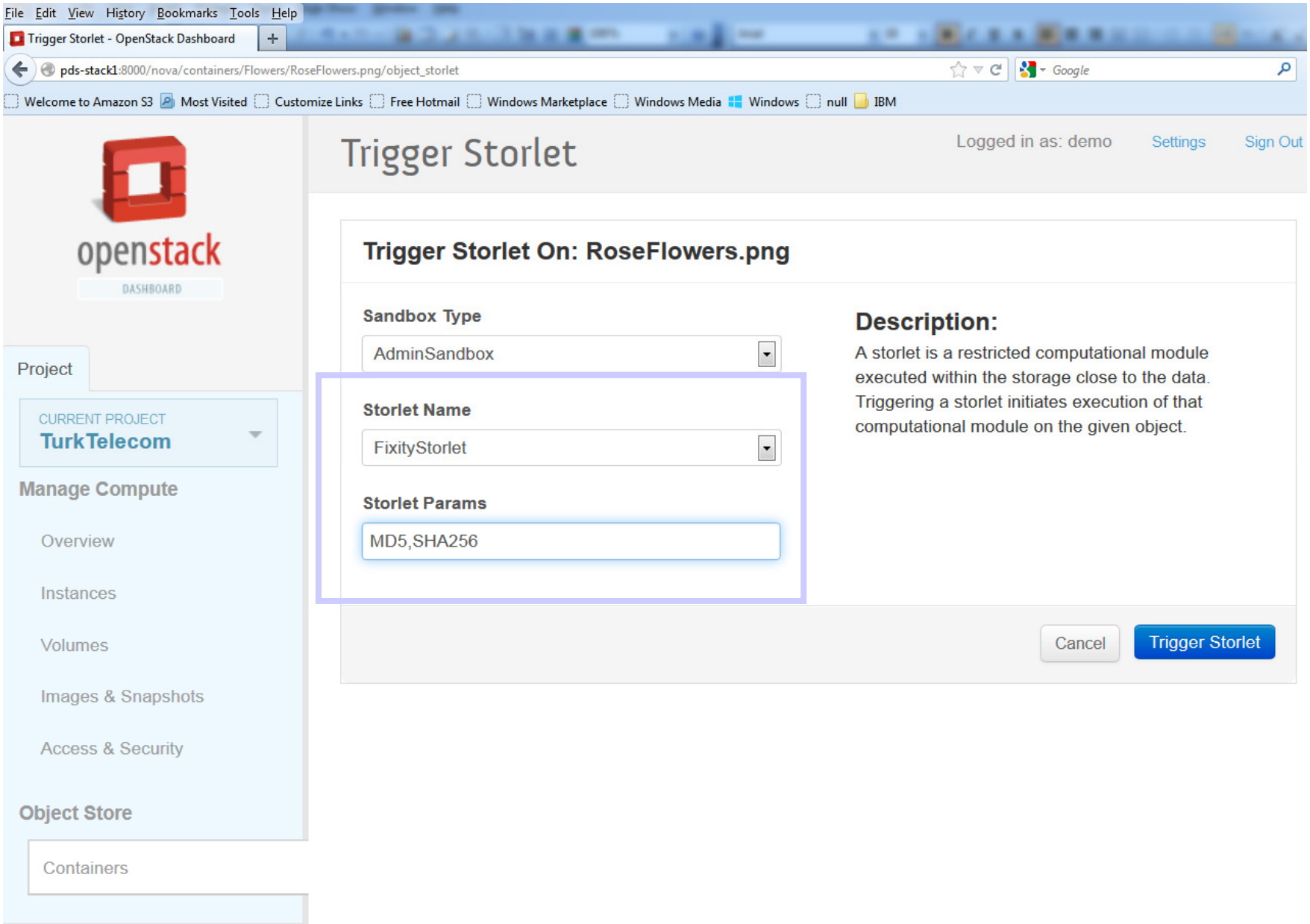
ROI Extraction: Storlet Result on Mobile



Cell Detection: Distributed Storlet Execution



Integrity Check: Trigger Fixity Storlet



Integrity Check: Fixity Storlet Result